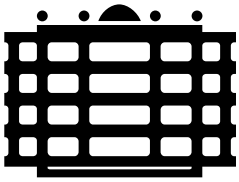

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

Professur Rechnernetze und verteilte Systeme

Diplomarbeit

Untersuchung und Bewertung von Netzzugangssteuerungen auf Basis des Standards 802.1x (Port-Based Network Access Control)

Verfasser:

Lars Richter

Betreuer:

Prof. Dr.-Ing. habil. Uwe Hübner

Jens Junghänel

Chemnitz, den 2. Januar 2005

Aufgabenstellung

Untersuchung und Bewertung von Netzzugangssteuerungen auf Basis des Standards 802.1x (Port-Based Network Access Control).

Zur Untersuchung gehört der Test von Netzwerkgeräten mit und ohne 802.1x Unterstützung. Im Zusammenhang dieser Untersuchungen soll eine auf Software basierende Authenticator-Komponente zum Einsatz kommen. Diese Komponente soll gegebenfalls aus dem *Open1x* Projekt adaptiert werden. Zusätzlich sollten Möglichkeit zum Test und zur Diagnose verteilter 802.1x/ Radius-Strukturen erarbeitet werden.
Im Rahmen dieser Arbeit sollte der Einsatz im Rechnernetz der Technischen Universität Chemnitz analysiert und eine Umgebung zur Netzzugangssteuerung mittels 802.1x geschaffen werden.

Erklärungen

Haftungsausschluss

Der Autor und die Technische Universität Chemnitz übernehmen keine Haftung für Schäden, welche sich aus der Benutzung dieser Arbeit oder der entwickelten Software ergeben.

Schutzrechte

Eingetragene Waren- und Markenzeichen sind in diesem Text nicht als solche gekennzeichnet. Das Fehlen der Kennzeichnung bedeutet nicht, dass diese Zeichen frei verwendbar sind.

Lizenzerteilung

Die zu dieser Arbeit gehörende Software wird nach Bedingungen der GNU General Public License (GPL) Version 2 als freie Software lizenziert.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ausschließlich mit Hilfe der aufgeführten Quellen angefertigt habe. Diese Arbeit hat noch keiner anderen Prüfungsbehörde vorgelegen.

Chemnitz, den 2. Januar 2005

Lars Richter

Danksagung

An dieser Stelle möchte ich mich bei einigen Personen bedanken, ohne die das Fertigstellen dieser Arbeit nicht oder nur schwierig möglich gewesen wäre.
Zunächst geht mein Dank an Herrn Prof. Hübner für die herausfordernde Aufgabe und die konstruktiven Gespräche während der Bearbeitung. Ein weiterer Dank geht an Herrn Jens Junghänel für die Betreuung dieser Arbeit und die Organisation der nötigen Hardware. Ebenso danke ich ihm und den Mitarbeitern des Rechenzentrums bei der schnellen Hilfe zur Lösung hardwaretechnischer Probleme.
Ein spezielles Dankeschön geht an Herrn Gerold Richter für das Durchsehen und die kritische Kontrolle dieser Arbeit. Weiterhin gilt mein Dank Herrn René Richter für seine TEXischen Hinweise.
Abschließend möchte ich mich noch bei denen bedanken, welche mich bewusst oder unbewusst während der Bearbeitung unterstützten. Nicht vergessen möchte ich alle, die mich täglich fragten, wann denn die Arbeit fertig ist - Jetzt ist sie es.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
CHAP	Challenge-Handshake Authentication Protocol
CGI	Common Gateway Interface
CRC	Cycle Redundant Check
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EAP	Extensible Authentication Protocol
EKU	Enhanced Key Usage
EAPOL	Extensible Authentication Protocol over LAN
EAPOW	Extensible Authentication Protocol over WLAN
FTP	File Transfer Protocol
GTC	Generic Token Card
GSM	Global System for Mobile Communication
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMAP	Internet Mail Access Protocol
IV	Initialisation Vector
IPSec	Internet Protocol Security
LAN	Local Area Network
LEAP	Lightweight EAP
MAC	Media Access Control
MD5	Message Digest 5
MIB	Management Informationbasis
MIC	Message Integrity Check
MPPE	Microsoft Point-to-Point Encryption
NTP	Network Time Protocol
OTP	One-Time Password
QoS	Quality of Service

PAP	Password Authentication Protocol
PEAP	Protected EAP
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
RADIUS	Remote Authentication Dial-In User Service
RSN	Robust Security Network
SNMP	Simple Network Management Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security
TLS	Tunneld TLS
TKIP	Tempolral Key Integrity Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunication System
VLAN	Virtual LAN
VoIP	Voice over Internet Protocol
VPN	Virtual Privat Network
WEP	Wired Equivalent Privacy
WLAN	Wireless LAN
WPA	Wi-Fi Protected Access
WPA-PSK	WPA-Pre-Shared Key
ZIN	Zertifikat der Internet Nutzung

Kapitel 1

Einleitung

Durch das ständige Wachstum an Netzwerken wächst auch stetig der Wunsch, diese immer besser abzusichern. Dieser Wunsch entsteht, da es immer mehr freie Zugangspunkte in existierenden Netzwerken gibt und ständig neue öffentliche Netzwerke entstehen, die mit persönlichen Geräten wie Laptop oder ähnlichen genutzt werden können. Beispiele für solche freien Netzwerke sind in Flughäfen, Schulen und Universitäten zu finden.
Ein wichtiger Mechanismus zur Gewährleistung der Sicherheit ist ein gutes Authentifizierungsverfahren, um sicher zu stellen, dass nur authorisierte Nutzer Zugriff besitzen. Jedoch ist die reine Anwendung eines solchen Verfahrens nicht sicher genug. Es ist ebenso wichtig, Maßnahmen zur Sicherung der sensiblen Daten eines jeden Nutzers einzusetzen. Meist entstehen Lücken im System, welche Angreifer durch falsch gewählte Passwörter oder durch ein ungesichertes Senden von Nutzerdaten im Netzwerk ausnutzen können. Dies erfolgt zum Beispiel bei der Verwendung des File Transfer Protokolls (FTP) [1] oder bei Zugriffen auf Postfächer für den Mailempfang. Um dieses Problem zu beheben, gibt es Maßnahmen, diese Daten während der Versendung auf nahezu jeder Protokollebene des Netzwerkes zu sichern. So existieren zum Beispiel Verfahren zum Schutz der Daten durch Virtual Privat Networks oder durch das Verwenden von Transport Layer Security (TLS)/ Secure Socket Layer (SSL) [2] für Anwendungsprotokolle wie das Hypertext Transfer Protokoll (HTTP) [3] oder dem Internet Message Access Protokoll (IMAP) [4]. Ebenso bieten manche Netzwerke bereits eine Sicherung der physischen Schicht an. Zu diesen gehören Funknetzwerke nach dem Standard 802.11 [5].
Die Verwendung einzelner Verfahren erhöht zwar den Schutz vor unberechtigten Zugriffen auf die Netzwerkstruktur, jedoch besitzen diese meist auch Schwachpunkte. Erst nach der Kombination mehrerer Verfahren erreicht man die erwünschte Sicherheitsstufe.
Das Ziel dieser Arbeit ist es, einen Authentifizierungsmechanismus für den Zugriff auf Netzwerke zu analysieren. Der Standard 802.1x - Port-Based Network Access Control, welcher vom Institute of Electrical and Electronics Engineers (IEEE) verabschiedet wurde, definiert ein Verfahren zur sicheren Authentifizierung direkt an der Schnittstelle, mit der sich der Nutzer mit dem Netzwerk verbunden hat.

Das anschließende Kapitel 2 stellt Maßnahmen vor, welche zur Sicherung und Zugangskontrolle von Netzwerken eingesetzt werden können. Im folgenden Kapitel 3 wird der Standard 802.1x näher betrachtet, welcher immer mehr Einsatz in verschiedenen bereits existierenden Lösungen findet. Im Kapitel 4 werden Soft- und Hardwarelösungen zur Verwendung des Verfahrens vorgestellt. Das 5. Kapitel richtet sich auf das Hauptziel dieser Arbeit. Es beinhaltet die Analyse für den Einsatz dieses Verfahrens im Rechnernetzwerk der Technischen Universität Chemnitz. Im Zusammenhang mit diesem Einsatz wurde eine Softwarelösung entwickelt. Diese wird im Kapitel 6 vorgestellt. Folgend an dieses sollen kurz noch Möglichkeiten zum Test und zur Diagnose dieses Verfahrens erörtert werden. Abschließend erfolgt eine Bewertung des Standards 802.1x im Zusammenhang mit der Nutzung im Rechnernetzwerk.

Kapitel 2

Netzwerkzugangsschutz

Es gibt eine Vielzahl an Möglichkeiten, Netzwerke vor unberechtigten Zugriffen zu sichern. Diese Verfahren existieren auf den meisten Ebenen des Protokollstacks. Einige Netzwerktechnologien besitzen bereits standardmäßig Möglichkeiten, die physische Datenübertragung zu sichern, und einige Technologien verfügen auch über eine sichere Zugangsauthentifizierung.
Das folgende Kapitel stellt einige ausgewählte Netzwerksicherungs- und Netzwerkzugangssysteme vor. Dazu soll der Unterschied zwischen Kabelnetzwerken und Funknetzwerken betrachtet werden, bevor abschließend einige mediumunabhängige Verfahren aufgezeigt werden.

2.1 Kabelnetzwerke

Kabelnetzwerke sind zur Zeit noch die am meisten verwendeten Netzwerke. Dabei gibt es Unterschiede in der Technologie. So existieren Bus-, Ring- und Sternstrukturen. Bei der Einführung beziehungsweise der Standardisierung dieser Netzwerke wurde ein Schutz der physischen Ebene nicht berücksichtigt. Ebenso existiert keine Möglichkeit, standardmäßig Nutzerauthentifizierungen durchzuführen. Dies war zum Zeitpunkt der Einführung nicht wichtig, da die Netzwerke meist nur in Forschungseinrichtungen und Einzelfirmen genutzt wurden. Auf Grund dieser anfänglichen kleinen Nutzung wurden, wenn überhaupt, lediglich Verfahren höherer Schichten zur Sicherung der Daten genutzt. Ebenso war der Anteil an frei zugänglichen Netzwerken gering, so dass keine explizite Zugangskontrolle benötigt wurde. Durch das schnelle Wachstum dieser Netzwerke entstand der Wunsch, Netzwerkzugriffspunkte frei verfügbar zu machen, um dem Nutzer das Verwenden eigener Hardware zu ermöglichen. Dies erforderte die Entwicklung von Zugangsmechanismen. Dazu wurden webbasierte Verfahren genutzt, welche zum Beispiel die Firewall zwischen dem zugänglichen Teil und dem internen Netzwerk nach erfolgter Anmeldung für den Nutzer passierbar machen. Ein weiterer eingesetzter Mechanismus ist die Verwendung von Virtual Privat Networks. In den letzten Jahren kamen Zugangssteuerungen durch 802.1x - Port-Based Network Access Control hinzu. Dieses neue Verfahren wird im folgenden Dokument noch näher betrachtet.

Durch diese entwickelten Mechanismen konnten Zugangssteuerungen realisiert werden. Zu beachten ist jedoch, dass eine Sicherung der Daten meistens nur zwischen dem Klient und dem Authentifizierungsserver besteht. Für einen weitergehenden Schutz bei der Kommunikation nach der Anmeldung über das Netzwerk ist der Nutzer selbst verantwortlich. Ihm stehen dazu weitere Sicherungsmöglichkeiten für die meisten Anwendungsprotokolle zur Verfügung.

2.2 Funknetzwerke nach 802.11

Im Gegensatz zu Kabelnetzwerken existieren in Funknetzwerken bereits Möglichkeiten, nur authorisierte Nutzer zuzulassen und deren Daten beim Senden zu sichern. Dies hängt mit dem Medium für die Datenübertragung zusammen. Der Zugriff auf Funknetzwerke ist nicht wie bei Kabelnetzwerken auf definierte Zugriffspunkte beschränkt. Das Mitlauschen nicht verschlüsselter Daten ist dadurch einfach zu erreichen. Aus diesem Grund wurden schon bei der Verabschiedung des Standards 802.11 für die Funknetzwerke Mechanismen für eine Sicherung einbezogen, welche in den folgenden Jahren verbessert wurden.
Die folgenden Teilabschnitte beschäftigen sich mit dem durch die Standardisierung entstandenen Sicherheitsverfahren. Diese sollen kurz vorgestellt und ihre Arbeitsweise und Probleme erörtert werden. Ein wichtiger Punkt ist der Überblick über die große Anzahl an Abkürzungen, welche im Zusammenhang mit verschiedenen Standardisierungen entstanden sind.

2.2.1 Wired Equivalent Privacy

Das erste Verfahren, welches zum Schutz entwickelt wurde, ist bekannt unter dem Namen Wired Equivalent Privacy (WEP) [6, 7]. Dieses Verfahren wurde bereits bei der Festlegung grundsätzlicher Eigenschaften von Funknetzwerken nach 802.11 [5] standardisiert. Es bietet die Möglichkeit, Daten zwischen dem Access Point und dem Nutzer durch Verschlüsselung zu sichern. Ebenso ist es nur den Nutzern möglich, das Netzwerk zu nutzen, welche den entsprechenden Schlüssel kennen. Diese Zugangssteuerung wird als Shared-Key Authentication bezeichnet. Die eigentliche Verschlüsselung der Folgedaten wird mit Hilfe des RC4-Algorithmusses durchgeführt. Hierbei handelt es sich um einen Stromchiffre, welcher durch die einfache logische Exclusiv-Oder Verknüpfung die Daten mit einem Geheimnis (Schlüssel) verschlüsselt. Bei WEP kommen Schlüssel mit einer Länge von anfangs 64bit und später 128bit zum Einsatz, wobei die eigentliche Schlüssellänge jeweils 24bit geringer ist. Diese fehlenden 24bit werden für den Initialisierungsverktor genutzt, welcher im Paket mit gesendet wird. Die Abbildung 2.1 zeigt die Verschlüsselung mittels WEP.
Zu Sicherung der Datenintegrität wird ein Cycle-Redundant Check (CRC-32) eingesetzt. Dieser erkennt zwar Veränderungen im Paket, gilt aber nicht als sicher, da dieser kein Geheimnis einbezieht. Hinzu kommt, dass diese Sicherung nur für die Nutz- und nicht für die Headerdaten erzeugt wird. Im folgenden wurden schnell die Sicherheitsprobleme von WEP erkannt. Es ist möglich, den verwendeten Netzwerkschlüssel durch Sammeln von Paketen einfach zu ermitteln. Das Verfahren zum

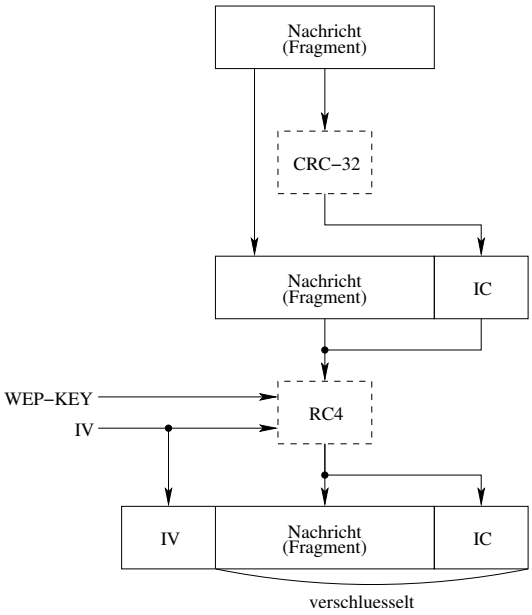

Abbildung 2.1: WEP - Verschlüsselung

Brechen des Schlüssel wird in [11] detaillierter vorgestellt. Diese Unsicherheit wird jedoch nicht vom Verschlüsselungsalgorithmus hervorgerufen, sondern entsteht erstens durch eine unzureichende Implementierung der Inititalisierungsvektoren und zweites durch bekannte Klartexte in den einzelnen Paketen. Erst diese Probleme machen das Brechen des Schlüssels möglich. Weiterhin ist die Sicherung der Datenintegrität problematisch. Da diese, wie erwähnt, nicht für die Headerdaten angewandt wird, können Angreifer gefälschte Pakete einspielen und dadurch Angriffe auf der Verbindungsschicht durchführen.
Die Sicherheit von WEP kann zwar nicht erhöht werden, aber durch den Einsatz von 802.1x, welcher folgend noch ausgiebig vorgestellt wird, kann eine verbesserte Nutzerauthentifizierung realisiert und ein Übermitteln der genutzten Schlüssel ermöglicht werden. Dies hat zur Folge, dass nicht direkt der Schlüssel bekannt gegeben werden muss. Die meisten Access Points besitzen die Möglichkeit, mehrere Schlüssel zu verwenden. Diese Anzahl ist jedoch begrenzt, und es erfolgt spätestens bei einer Nutzeranzahl, welche um eins größer ist als die Schlüsselmenge, eine Mehrfachverwendung dieser. Die eigentlichen Probleme von WEP werden, wie bereits erwähnt, nicht behoben. So besteht weiterhin die Möglichkeit, dass Angreifer die gegebenen Schwächen ausnutzen und Sitzungen übernehmen oder sensible Daten nach dem Brechen des Schlüssels für einen späteren Gebrauch sammeln.

2.2.2 Wi-Fi Protected Access

Auf Grund der hohen Risiken beim Einsatz von WEP wurde begonnen, einen neuen Standard für die Sicherung und den Zugangsschutz zu erstellen. Da jedoch die Verabschiedung eines Standards in den meisten Fällen eine hohe Zeit beansprucht und der Druck zur Verbesserung der Sicher-

heit in der Funkttechnologie ständig wuchs, wurde die Wi-Fi Allianz gegründet. Diese bezog sich auf den im Entwicklungsstatus befindlichen 802.11i Standard und entnahm diesem einige wichtige Punkte zur Verbesserung der Sicherheit. Diese Punkte wurden unter dem Namen Wi-Fi Protected Access (WPA) [8, 9, 10] zusammengefasst. Der wichtigste Bestandteil dieser Zusammenfassung ist die Eliminierung der Schwachstellen von WEP. Zusätzlich wurden die Eigenschaften zur Integritätssicherung verbessert und ebenso eine Authentifizierung eingeführt. Bei der Umsetzung war es wichtig, diese Neuerungen auch bei Verwendung alter Hardware effizient nutzbar zu machen.
Das durch WPA eingeführte Temporal Key Integrity Protocol (TKIP) fügt dem WEP-Verfahren folgende neuen Bestandteile hinzu. Die Sicherung der Datenintegrität erfolgt nicht mehr durch einen einfachen CRC-32. Es wird ein neuer Algorithmus namens Michael für den Message Integrity Code (MIC) verwendet. Dieser benötigt ebenfalls einen Schlüssel. Die Integritätssicherung wird auf die Headerdaten der Pakete erweitert. So soll verhindert werden, dass diese gefälscht und für spätere Angriffe genutzt werden. Ein Zähler im MIC dient der Ermittlung fehlerhafter Pakete. Ist eine definierte Anzahl an falschen oder fehlerhaften Paketen in einer definierten Zeitperiode eingetroffen, so nimmt der Access Point an, dass die Verbindung unter einem Angriff steht und bricht diese ab. Anschließend ist eine neue Anmeldung erforderlich. Zusätzlich zu dieser Verbesserung wurden dem eigentlichen Verschlüsselungsverfahren neue Bestandteile hinzugefügt. Es wurden so genannte "Per-Paket Keys" eingeführt. Das heißt, dass jedes Paket mit einem unterschiedlichen Schlüssel verschlüsselt wird. Diese Schlüssel werden aus mehreren verschiedenen Parametern abgeleitet. Die Grundlage dafür bilden Schlüssel, welche bei der Authentifizierung ausgehandelt wurden. Aus diesen werden der "Temporal Key" und der Schlüssel für den MIC abgeleitet. Mit Hilfe des "Temporal Key" und der Hinzunahme der MAC-Adresse des Senders und einer Sequenznummer wird der "Per-Paket Key" erzeugt. Als Sequenzzähler wird der Initialisierungsvektor verwendet. Dieser wird nach dem Setzen neuer Schlüssel beim Sender und Empfänger auf null initialisiert. Mit Hilfe dieses Sequenzzähler können so genannte Replay-Attacks, das heißt, das Einspielen alter Pakete verhindert werden. Durch das Erzeugen von "Per-Paket Keys" besteht nicht nur eine Trennung der einzelnen Verbindungen, sondern es werden auch schwache Schlüssel vermieden. Die Abbildung 2.2 zeigt das erweiterte Verfahren zur Verschlüsselung. Im Kern dieser Abbildung ist die ursprüngliche Architektur von WEP zu erkennen, welche durch entsprechende Funktionalität zur Sicherung erweitert wurde.
Die Abbildung zeigt, dass der MIC nicht nur für ein Datenfragment, sondern über die gesamte Nachricht unter Einbeziehung der Quell- und Zieladresse berechnet wird. Der Empfänger kann diesen somit erst nach dem Erhalt der vollständigen Nachricht prüfen. Das Erzeugen der Schlüssel erfolgt zweistufig. Die erste Stufe muss nicht bei jedem Paket neu berechnet werden, da die Elemente nicht ständigen Änderungen unterliegen. Erst bei hohen Sequenznummern oder bei der Änderung der temporären Schlüssel erfolgt diese Berechnung neu. Die zweite Stufe muss bei jedem Paket neu durchlaufen werden. Die Änderung unterscheidet sich jedoch nur im Sequenzzähler, welcher inkrementiert wird. Dadurch ist eine Vorrausberechnung dieser Schlüssel möglich, um Zeit zu sparen.

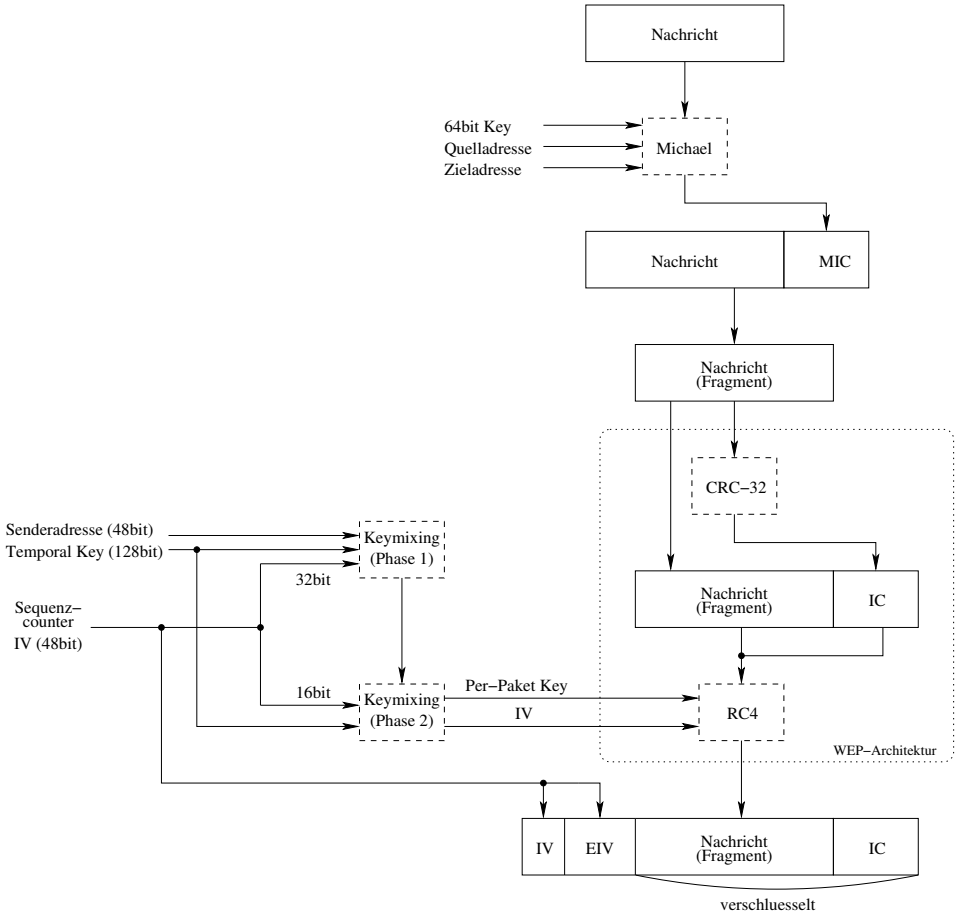

Abbildung 2.2: WPA - Verschlüsselung

Das Ergebnis ist ein 128bit langer Schlüssel für den RC4-Algorithmus.
Eine weitere Neuerung bei WPA ist die Forderung nach einer Authentifizierung. Diese wird durch das später noch vorgestellte Verfahren von 802.1x erreicht. Mit Hilfe dieses Verfahrens werden die nötigen Ausgangsschlüssel, welche später der Ableitung für den MIC Schlüssel und den "Temporal Key" dienen, erzeugt, wobei dies stark von der genutzten Authentifizierungsmethode abhängt. Da für diesen Zweck weitere Servertechnologien benötigt werden, ist die Authentifizierung auch mittels eines Pre-Shared Secrets (WPA-PSK) möglich. Dabei werden die Schlüssel nicht durch die Authentifizierung erzeugt, sondern aus dem fest definierten Geheimnis des Access Points abgeleitet. Dieses Geheimnis muss dem Klienten zur Verfügung gestellt werden. Ein Rekeyingmechanismus verhindert das zu lange Nutzen der erzeugten Ausgangsschlüssel, was einen zusätzlichen Schwachpunkt von WEP eliminiert.
Diese Neuerungen erhöhen die Sicherheit von Funkverbindungen. Durch einfache Firmwareupdates kann in den meisten Geräten die zusätzlichen Funktionalität bereitgestellt werden. Jedoch bringen diese Änderungen auch einige Probleme mit sich, auf die bei der Umsetzung zu Gunsten der Nutzung alter Hardware nicht geachtet wurde. So ist es durch fehlende Managementfunktionen dem Access Point nicht möglich, ein Roaming ohne erneute vollständige Authentifizierung durch-

zuführen. Diese erneute Authentifizierung erfordert bei der Nutzung von Echtzeitanwendungen, wie zum Beispiel Voice over IP (VoIP), zu viel Zeit. Ein zusätzlicher Punkt ist Vernachlässigung der durch den Standard 802.11 definierten Quality of Service (QoS). Weiterhin besteht in Ad-hoc Umgebung keine gültige Session, was wiederum bei jeder Verbindung eine Authentifizierung erfordert. In Umgebungen, wo diese Parameter nicht entscheidend sind, zählt WPA zu der sichersten und am einfachsten zu konfigurierenden Methode.

2.2.3 Standard 802.11i

Der neue Standard 802.11i [12, 10] zur Sicherung von drahtlosen Verbindung wurde im Juni 2004 verabschiedet. Dieser bringt zusätzliche Neuerungen zur Sicherung mit sich. Diese zielen speziell auf die Ersetzung des RC4-Algorithmusses und einer Verbesserung der Datenintegritätssicherung. Die erfolgten Änderungen können meistens nicht durch ein einfaches Firmwareupdate auf alter Hardware verfügbar gemacht werden, da die grundlegende Hardwarestruktur für die neuen Methoden zu langsam ist.
Der Standard 802.11i, welcher bei der Wi-Fi Allianz den Namen WPA2 trägt, definiert eine neue Netzwerkarchitektur. Diese wird als Robust Security Network (RSN) bezeichnet und soll den zukünftigen Sicherheitsstandard für Funknetzwerke darstellen.
Die Grundlage dieser Architektur bildet ein Adäquat zu dem bei WPA verwendeten TKIP. Die verbesserte Version baut nicht mehr auf die Nutzung von RC4 und Michael auf, sondern verwendet den Advanced Encryption Standard (AES) im so genannten Counter-Mode (CTR) und zur Sicherung der Datenintegrität den Cipher-Block-Chaining Message Authenication Code (CBC-MAC). Die beiden Bestandteile wurden unter dem Namen CTR/CBC-MAC Protokoll (CCMP) zusammengefasst. Diese Methode benötigt zur Verschlüsselung und zur Integritätssicherung in Gegensatz zu WPA lediglich einen Schlüssel, welcher durch eine Länge von 128bit definiert ist. Ebenso wie bei WPA wird eine Authentifizierung und ein Schlüsselmanagement mittels 802.1x durchgeführt, wobei wiederum nicht alle Authentifizierungsverfahren möglich sind. Die Sicherheit von Funknetzwerken wurde durch das Ersetzen der Kernstrukturen drastisch verstärkt. Das Prinzip zum Schutz ist jedoch gleich dem vom WPA. So existiert ebenfalls ein Sequenzzähler, welcher vor Replay-Attacks schützt und ein MIC, um gefälschte Pakete zu erkennen. Zusätzliche Änderungen gegenüber WPA sind eine schelle Reauthentifizierung, um ein schnelles Roaming zu erreichen, und ein geschützter Ad-hoc Modus. Damit werden die Probleme, welche noch bei WPA existieren, behoben.
Die folgende Abbildung 2.3 zeigt den schematischen Ablauf der Verschlüsselung mittels 802.11i.

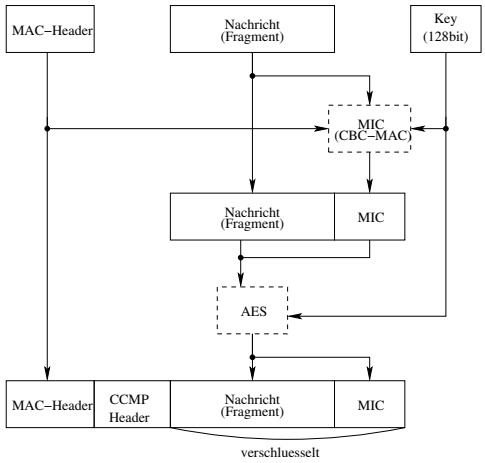

Abbildung 2.3: 802.11i - Verschlüsselung

2.2.4 Zusammenfassung

In diesem Abschnitt soll eine kurze tabellarische Übersicht die drei Verfahren (WEP/ WPA/802.11i) zusammenfassen. In der Tabelle 2.2.4 werden die verwendeten Methoden und die verschiedenen Parameter gegenübergestellt.

Verfahren	**WEP**	**WPA**	**802.11i**
Verschlüsselung	RC4	RC4	AES-CTR
Schlüssellänge	64/128bit	128bit	128bit
IV-Länge	24bit	48bit	48bit
Datenintegrität	CRC-32	Michael	CBC-MAC
Headerintegrität	keine	Michael	CBC-MAC
Authentifizierung	Shared Key	802.1x	802.1x
Keymanagement	keins	802.1x	802.1x
Replay-Attack Sicherung	keine	IV-Sequenz	IV-Sequenz

Tabelle 2.1: Sicherheitsverfahren für 802.11 Netzwerke

Mit Hilfe von WPA und des neuen Standards 802.11i lassen sich sichere Funknetzwerke erstellen. Diese besitzen die Möglichkeit, mit Hilfe von Authentifizierungsmechanismen Zugangssteuerungen bereitzustellen. Ein mittels WEP geschütztes Netzwerk kann dies standardmäßig nicht. Es müssen dazu weitere Maßnahmen getroffen werden. Die Einführung von 802.1x behebt zwar das Fehlen der Nutzerauthentifikation, jedoch nicht den fehlenden Schutz der Pakete im Funknetzwerk. Aus diesem Grund sollte auf den Einsatz von WEP bei Funknetzwerken verzichtet werden. Ebenso ist zu beachten, dass dieser durch die einzelnen Verfahren erzielte Schutz durch Verschlüsselung der Daten nur zwischen Access Point und Klient besteht. Für eine weitergehende Sicherung muss der Nutzer selbst sorgen.

2.3 Mediumunabhängige Zugangsschutzlösungen

Zusätzlich zu dem schon erwähnten Verfahren 802.1x oder Pre-Shared Keys für Funknetzwerke gibt es weitere Möglichkeiten, ein Zugangsmanagement bereitzustellen. Es gibt Verfahren, welche nicht nur eine Authentifikation von Nutzern durchführen, sondern auch eine Sicherung, das heißt, eine Verschlüsselung der Folgedaten auf Teilstücken des Netzwerkes bewirken. Diese weiteren Verfahren unterscheiden sich lediglich in der Ansiedlung im Protokollstack. Das Problem besteht darin, je höher ein Verfahren wirksam wird, desto mehr Informationen müssen dem Nutzer schon vor der Anmeldung bekannt sein. Die Tabelle 2.3 zeigt das ISO-OSI Referenzmodel [13] mit einigen ausgewählten Verfahren zur Sicherung von Netzwerken.

Schicht	**Verfahren**	**Referenz**
Anwendung	HTTP/HTTPS (webbasiert)	[3] [2]
Präsentation		
Sitzung		
Transport		
Netzwerk	IPSec	[14, 15]
Verbindung	802.1x, PPTP	[16] [17, 18]
physisch		

Tabelle 2.2: Sicherheitsverfahren und ihre Ansiedlung

Da solche Verfahren nicht an die physische Schicht gebunden sind wie die für Funknetzwerke standardisierten Varianten, können die im folgenden kurz vorgestellten Verfahren in allen Netzwerken eingesetzt werden.
Ein geeignetes Verfahren ist das Einsetzen von Virtual Privat Networks (VPN). Diese wurden zum Zugriff auf interne Netzwerke von Firmen oder Institutionen aus dem Internet entwickelt. Da das Medium Internet als potenziell unsicher angenommen wird, wird mit Hilfe von VPN ein Tunnel zwischen dem Anwender und dem Zielnetzwerk hergestellt und durch Verschlüsselungsverfahren gesichert. Mit Hilfe dieser Technologie lässt sich ebenso ein Zugangssystem für öffentliche Netzwerke realisieren. Alle öffentlichen Teile befinden sich in einem potentiell unsicheren Anmeldenetzwerk. Nach erfolgtem Aufbau einer VPN-Verbindung wird ein Tunnel durch das Anmeldenetzwerk in das interne Netz hergestellt. Ein Nachteil bei diesem Verfahren ist der Einsatz zusätzlicher Software. Ein weiteres Problem besteht darin, dass Angriffe auf tiefer liegenden Ebenen des Protokollstacks weiterhin durchführbar sind. Zu den am meisten eingesetzten Vertretern dieser Technologie gehören das Point-to-Point Tunneling Protocol (PPTP) und Internet Protocol Security (IPSec).
Ein weiteres eingesetztes Verfahren ist die Bereitstellung eines Zugangssystems über eine Web-Schnittstelle. Dies erfolgt mit Hilfe eines Webservers im Anmeldenetzwerk. Er nimmt die nötigen Anmeldedaten des Nutzers entgegen, prüft diese und schaltet den Nutzer durch gezieltes Setzen von

Firewallregeln für das interne Netzwerk frei. Die Anmeldung erfolgt meist über einen gesicherten Kanal, welcher mittels Transport Layer Security (TLS)/ Secure Socket Layer (SSL) erzeugt wird. Da dieses Verfahren auf einer sehr hohen Ebene des Protokollstacks arbeitet, benötigt es zusätzliche Servertechnologie, welche gewartet und administriert werden muss. Ebenso sind die nötigen Informationen, welche der Nutzer vor der Anmeldung besitzen muss, relativ hoch. Dadurch entstehen auch eine Vielzahl von Angriffsmölichkeiten. Ein Vorteil ist, dass auf Nutzerseite meist keine zusätzlichen Programme verfügbar sein müssen. Es ist nur ein Webbrowser und gegebenfalls ein Konfigurationsprogramm für dynamische Zustellung von Netzwerkadressen notwendig. Diese sind in den meisten Betriebssystemen standardmäßig vorhanden. Ein weiterer Vorteil ist, dass sich diese Systeme in den meisten Netzwerkstrukturen umsetzen lassen. Durch eigene Implementationen lassen sich auch zusätzliche Sicherheitsmaßnahmen realisieren.

Kapitel 3

Standard 802.1x - Port-Based Network Access Control

Der Standard 802.1x - Port-Based Network Access Control [16] wurde zur Authentifizierung in Netzwerken entwickelt. Diese erfolgt direkt am Eintrittspunkt, mit dem sich der Nutzer verbunden hat. Dazu zählen zum Beispiel die Ports eines Switches oder in Funknetzwerken der für den Zugriff nötige Access Point. Der Vorteil dieses Verfahrens ist, dass die Schnittstelle erst nach einer Authentifizierung freigeschalten wird. Es ist somit nicht möglich, eine Kommunikation ohne Authentifizierung mit anderen Netzwerkteilnehmern zu etablieren, da unauthentifizierte Schnittstellen keinen Datentransfer außer den für die Authentifizierung notwendigen zulassen.
Dieses Verfahren entstand aus einem bereits für Point-to-Point Verbindungen [19] existierenden Authentifizierungsverfahren. Dort wurde das Extensible Authentication Protocol (EAP) zur Authentifizierung verwendet, welches beim Standard 802.1x ebenfalls die Grundlage der Authentifizierung ist. Im Gegensatz zum Verfahren bei Point-to-Point Verbindungen mussten für lokale Netzwerke einige Änderungen vorgenommen werden.
Im anschließenden Kapitel wird dieser Standard näher beschrieben. Dazu werden die Änderungen gegenüber den Point-to-Point Verbindungen betrachtet. Zusätzlich werden der Ablauf einer Authentifizierung mit den daran Beteiligten sowie die verwendeten Netzwerkprotokolle und die Authentifizierungsverfahren detailliert erläutert. Ebenfalls wird auf die Sicherheit des durch den Standard 802.1x beschriebenen Verfahrens eingegangen.

3.1 Authentifizierungsteilnehmer

Die Abbildung 3.1 zeigt ein Netzwerk, welches die Möglichkeit bietet, sich bei der drahtlosen Verbindung über das Verfahren von 802.1x zu authentifizieren.
Der Klient, welcher Zugang zum System erhalten möchte, wird als ***Supplicant*** bezeichnet. Er stellt eine Verbindung zum ***Authenticator*** her. Dieser ist im dargestellten Szenario der Access Point für das drahtlose Netz. Die Authentifizierung wird durch den ***Authentication-Server*** realisiert, wel-

cher sich im internen Teil des Netzwerkes befindet. Der Authenticator besitzt keinerlei Information über Nutzer und deren Authentifizierungsdaten. Er dient als Ansprechpartner für den Supplicanten und ermöglicht eine Kommunikation mit dem Authentication-Server. Da der Supplicant auf physischer Ebene einen direkten Ansprechpartner besitzt, benötigt er keinerlei Informationen über das Netzwerk, mit welchem er sich verbunden hat.

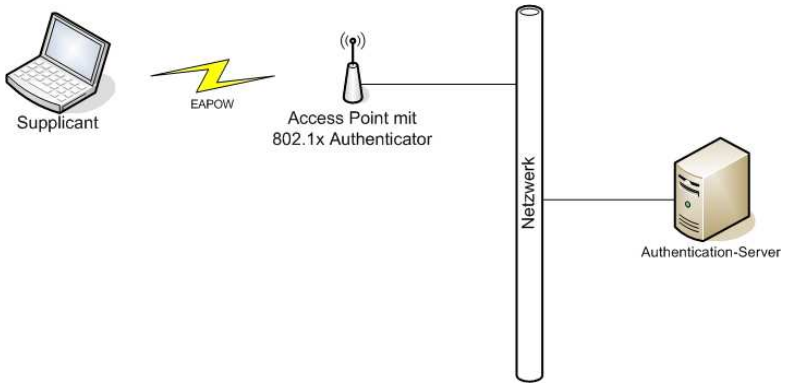

Abbildung 3.1: Netzwerkszenario mit 802.1x Authenticator

Dieses Verfahren kann in drahtlosen und drahtgebundenen Netzwerken verwendet werden. Wichtig ist, dass der Authenticator einen Zugang zum Autentication-Server besitzt, um die Authentifizierung durchzuführen und ständig eine Kommunikation zwischen dem Supplicanten und dem Authenticator möglich ist.

3.2 Protokolle

Die Kommunikation während der Authentifizierung zwischen Supplicant und Authenticator erfolgt über die Verbindungsschicht des ISO-OSI Referenzmodelles. Dabei werden zwei Protokolle verwendet, welche folgend näher betrachtet werden. Zusätzlich besitzen die Pakete der Verbindungsschicht besondere Eigenschaften, welche für die Übertragung der Daten notwendig sind. Diese Eigenschaften werden ebenfalls im nächsten Abschnitt näher erläutert, bevor die für die Authentifizierung notwendigen Protokolle betrachtet werden.

3.2.1 Verbindungsschicht

Für die Übertragung der Pakete auf dieser Schicht wurde ein spezieller Pakettyp (siehe Tabelle 3.1) und eine Zieladresse (siehe Tabelle 3.2) definiert. Die Zieladresse ist eine Multicast-Adresse, da in drahtgebundenen Netzwerken die Adresse des Authenticators nicht bekannt ist. In drahtlosen Netzwerken existiert dieses Problem nicht, da der Authenticator meist direkt im Access Point integriert und der Kommunikationsweg zwischen Supplicant und Authenticator durch das Verbinden mit dem Funknetzwerk bekannt ist.

Als weitere Eigenschaft dieser Multicast-Adresse muss beachtet werden, dass diese nicht durch Switches weitergeleitet wird. Diese Adresse ist nach dem Standard 802.1d [20] als "Bridge Group Address" definiert und unterliegt nach den Vorgaben dieses Standards einer statischen Filterung im Gerät. Das bedeutet, die Nutzung dieses Verfahrens mit der Verwendung der Multicast-Adresse in verteilten Netzwerken, welche durch Switches realisiert werden, ist nicht möglich. Das Problem kann jedoch umgangen werden, indem die direkte Adresse des Authenticators in den Paketen als Zieladresse enthalten ist. Diese Einstellung der Anpassung der Zieladresse erfolgt direkt im Supplicanten. Die Verwendung einer Unicast-Adresse stellt allerdings ein erhöhtes Sicherheitsrisiko dar. Ein Angreifer könnte zum Beispiel durch gefälschte Pakete bereits authentifizierte Nutzer abmelden und somit einen Denial-of-Service Angriff durchführen. Der Abschnitt Abschnitt 3.5 auf Seite 23 gibt Aufschluss über die Sicherheitsrisiken, welche in bestimmten Situationen auftreten können.

Pakettyp der Verbindungsschicht	0x888E

Tabelle 3.1: Pakettyp der Verbindungsschicht

Multicast-Adresse der Verbindungsschicht	01:80:C2:00:00:03

Tabelle 3.2: Multicast-Adresse der Verbindungsschicht

3.2.2 Extensible Authentication Protocol

Wie bereits erwähnt, wurde dieses Protokoll für die Authentifizierung von Point-to-Point Verbindungen entwickelt und komplett in den Standard 802.1x übernommen. Dieses Verfahren wird in [21] detailliert beschrieben.
Der Aufbau der EAP Pakete ist leicht überschaubar, wie die Abbildung 3.2 zeigt. Der Eintrag des EAP-Code Feldes kann lediglich vier verschiedene Werte annehmen. Diese sind "Request", "Response", "Success" und "Failure". Das Identifier Feld ist eine eindeutige Zahl, welche bei jedem neuen Paket hochgezählt wird. Lediglich ein "Response" Paket verwendet den Identifier des vorher gesendeten "Request" Paketes, um eine Zuordnung zu erhalten und Wiederholungssendungen zu erkennen.
Pakete der Art "Request" und "Response" besitzen Paketdaten, welche sich in zwei Teile gliedern. Zuerst wird der Datentyp übermittelt, welcher angibt, wie die folgenden Daten behandelt werden sollen. Eine Liste der möglichen Werte für den Datentyp befindet sich im Anhang A.1.2 auf Seite 68. Bei den beiden anderen Paketarten werden keine zusätzliche Daten übermittelt.

EAP – Code (1 Byte)
EAP – Identifier (1 Byte)
EAP – Data Length (2 Byte)
EAP – Data (n Byte)

Abbildung 3.2: EAP - Paketformat

Das Extentsible Authentication Protocol (EAP) wird zum Transport der Authentifizierungsdaten verwendet. Da es aber nicht für das Initiieren oder Beenden einer Verbindung verwendet werden kann, wurde speziell zur Nutzung dieses Verfahrens bei 802.1x ein Rahmenprotokoll definiert, welches im nächsten Abschnitt näher betrachtet wird.

3.2.3 Extensible Authentication Protocol over LAN

Für dieses Protokoll existieren mehrere Bezeichnungen je nach Anwendungsgebiet. So heißt es in drahtgebunden Netzwerken Extensible Authentication Protocol over LAN (EAPOL) und in drahtlosen Netzwerken Extensible Authentication Protocol over WLAN (EAPOW). Es handelt sich jedoch um ein und das selbe Protokoll. Dieses Protokoll ist lediglich ein Rahmenprotokoll für das intern gekapselte EAP. Es wird verwendet, um durch den Supplicanten eine neue Sitzung zu initiieren, zu beenden oder die Daten des EAP zu transportieren. Ebenfalls wie beim EAP ist der Aufbau eines Paketes einfach gehalten. Abbildung 3.3 zeigt den Protokollkopf. Speziell für die Verwendung in Funknetzwerken existiert in diesem Protokoll ein zusätzlicher Mechanismus. Dieser wird später in diesem Abschnitt näher beschrieben.

EAPOL – Version (1 Byte)
EAPOL – Type (1 Byte)
EAPOL – Data Length (2 Byte)
EAPOL – Data (n Byte)

Abbildung 3.3: EAPOL - Paketformat

Die möglichen Arten von Paketen bei diesem Protokoll sind ebenfalls gering. So enthält das Feld Version standardmäßig den Wert "1", da momentan nur die Version 1 dieses Protokolls verabschiedet wurde. Der Eintrag der Datenlänge definiert die Länge der gekapselten Daten.

Der wichtigste Bestandteil ist das Typefeld, für das fünf verschiedene Werte definiert sind, die im folgendem aufgeführt sind.

- **EAPOL-EAP** (Wert: 0)
  - Kennzeichnung für ein gekapseltes EAP Paket
- **EAPOL-Start** (Wert: 1)
  - Initiierung der Verbindung zum Authenticator
- **EAPOL-Logoff** (Wert: 2)
  - Beenden der Verbindung zum Authenticator
- **EAPOL-Key** (Wert: 3)
  - Übermittlung eines Schlüssel für Funknetzwerke mit WEP/WPA/802.11i
- **EAPOL-Alert** (Wert: 4)
  - Übermittlung einer Fehlermeldung

Pakete mit dem "EAPOL-Start" oder "EAPOL-Logoff" Typ besitzen keinen zusätzlichen Dateninhalt. Diese Pakete werden benötigt, um eine Verbindung vom Supplicanten zum Authenticator zu initiieren, beziehungsweise nach Abschluss der Sitzung das Port, welches authentifiziert wurde, wieder in den nicht authentifizierten Zustand zu versetzen. Ein Paket vom Typ "EAPOL-Alert" dient der Übermittlung von Fehlermeldungen während der laufenden Sitzung zwischen Authenticator und Supplicant. Der Inhalt des Paketes ist dementsprechend eine Fehlermeldung, wobei für die Art des Aufbaus der Fehlermeldung keine Richtlinie existiert. Da es sich bei diesem Protokoll um ein Transportprotokoll für das EAP handelt, existiert ein entsprechender Type, welcher den Transport ersichtlich macht.
Ein wesentlicher Bestandteil in drahtlosen Netzwerken ist die Möglichkeit, "EAPOL-Key" Pakete zu versenden. Dies ist auch der Grund der unterschiedlichen Namensgebung, da diese Pakete nur bei Funknetzwerken sinnvoll sind. In drahtgebundenen Netzwerken existiert keine Punkt zu Punkt Verschlüsselung der physischen Schicht. Pakete dieses Typs werden deswegen zur Übermittlung von Schlüsseln für die einzelnen Verfahren von Funknetzwerken benutzt. Dadurch kann nicht nur eine dynamische Schlüsselvergabe erreicht werden, sondern auch nach Ablauf einer definierten Zeitperiode ein Ändern der Schlüssel durch einen Rekeyingmechanismus. Das Erzeugen der Schlüssel basiert auf Daten, welche während der Authentifizierung ausgetauscht wurden. Jedoch ist nicht jedes Authentifizierungsverfahren zur Generierung von Schlüsseln einsetzbar. Welche genutzt werden können und welche diese Funktionalität nicht bieten, wird im Abschnitt 3.4 auf Seite 19 genauer beschrieben.

3.3 Ablauf einer Authentifizierung

Eine Authentifizierung findet in mehreren Schritten statt. Supplicant, Authenticator und Authentication-Server sind nicht immer volle Teilnehmer der stattfindenden Kommunikation. In der Abbildung 3.4 ist der schematische Ablauf einer Authentifizierung mittels der Authentifizierungsmethode Transport Layer Security (TLS) dargestellt.

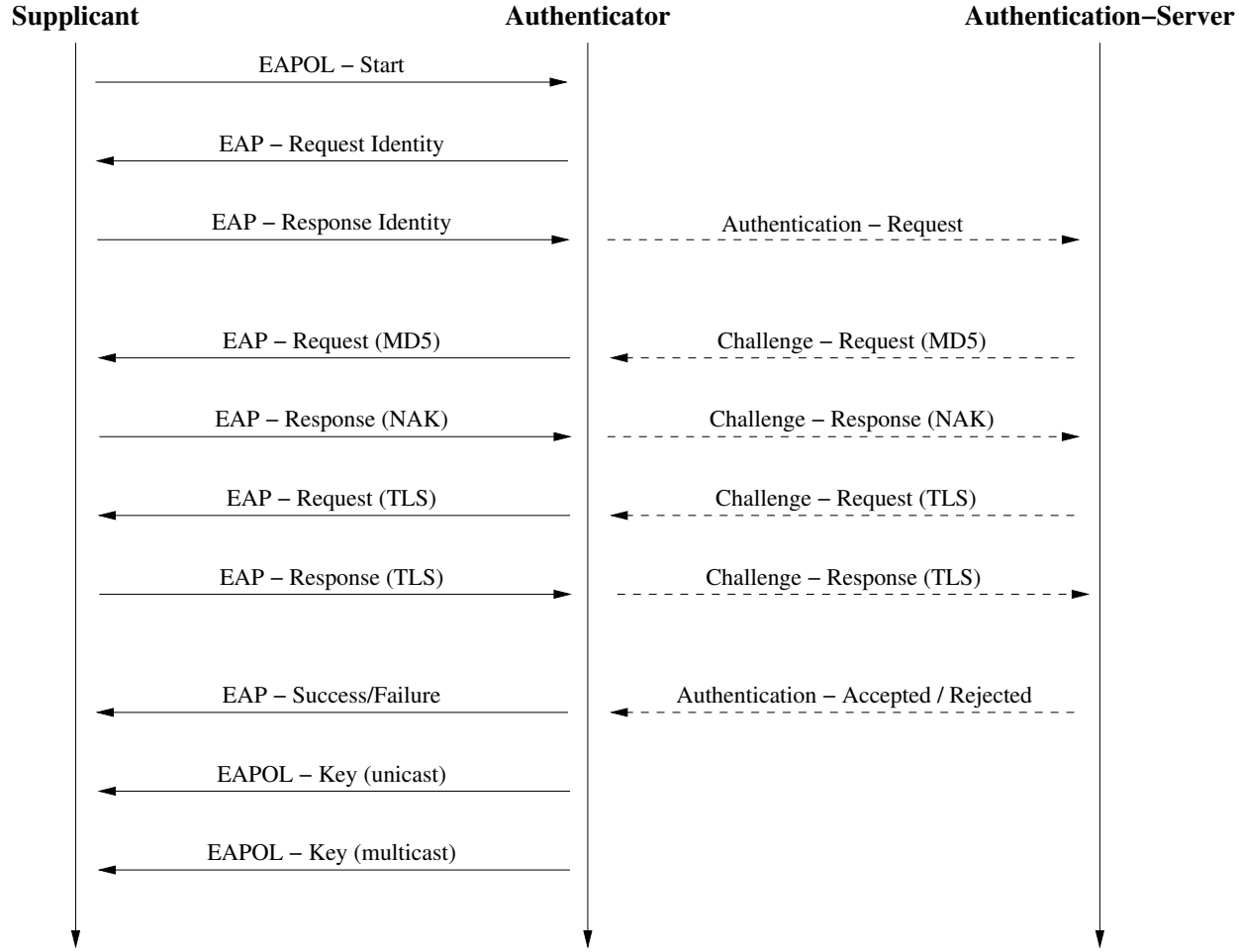

Abbildung 3.4: Authentifizierungsablauf

Die erste Phase beginnt durch das physische Verbinden des Supplicanten mit dem Netzwerk. Bei drahtgebunden Netzwerken erfolgt dies über das Anstecken des Kabels an ein im Netzwerk aktives Gerät zum Beispiel an einen Switch. In drahtlosen Netzwerken wird dies über die Verbindung zwischen Klient und Access Point erreicht. Ist die Verbindung mit dem Netzwerk erfolgreich hergestellt, sendet der Supplicant ein "EAPOL-Start" Paket und versucht damit, einen Authenticator zu erreichen. Sollte kein Authenticator antworten, bricht der Supplicant den Versuch der Authentifizierung ab und sieht das Netzwerk als freigegeben an.
Antwortet ein Authenticator, so beginnt die eigentliche Authentifizierungsphase. Der Authenticator sendet dem Supplicanten ein Paket, um dessen Identität zu erfahren. Der Supplicant antwortet entsprechend mit einem "EAP-Response" Paket und übermittelt seine Identität. Es gibt Authentifi-

zierungsverfahren, welche eine innere und eine äußere Identität besitzen. Diese werden im folgendem Abschnitt 3.4 näher erläutert. Nach Erhalt des Paketes vom Supplicanten sendet der Authenticator die Anfrage für eine Authentifizierung an den Authentication-Server. Dazu kapselt er das erhaltene EAP Paket in eins für den Authentication-Server verständliches Paket.
Ab dem jetzigen Zeitpunkt erfolgt die Kommunikation für die Authentifizierung zwischen Authencation-Server und Supplicant. Der Authenticator dient lediglich dem Wandeln der Pakete des Authentication-Server in reine EAP Pakete für den Supplicanten und umgekehrt. Er spielt in dieser Phase der Authentifizierung die Rolle eines Proxyservers. Der Authentication-Server sendet ein gekapseltes "EAP-Request" Paket, um das Verfahren der Authentifizierung festzulegen. In der Abbildung 3.4 sieht man die Forderung für das MD5-Verfahren. Da dies der Supplicant nicht unterstützt, folgt als Antwort ein "EAP-Response" Paket vom Typ "NAK". Dieser Typ kann nur in Response-Paketen verwendet werden und wird genutzt, um ein anderes Authentifizierungsverfahren zu erlangen. Danach wird der Authentication-Server versuchen, weitere Verfahren anzubieten. Erfolgt keine Übereinstimmung bei der Aushandlung des Verfahrens, wird dem Supplicant mittels eines "EAP-Failure" Paketes die Authentifizierung verwehrt. Andernfalls beginnt die Übermittlung der für dieses Verfahren benötigten Daten. Die Abbildung zeigt den Erfolg für ein Authentifizierungsverfahren mittels Transport Layer Security (TLS). Die Funktionsweise dieses Verfahrens ist in 3.4.2 näher erläutert.
Sind alle benötigten Daten zur Authentifizierung ausgetauscht, entscheidet der Authentication-Server über einen Erfolg oder Misserfolg der Authentifizierung. Bei einem Erfolg werden im Paket vom Authentication-Server notwendige Informationen zur eventuellen Schlüsselgenerierung für drahtlose Netzwerke mitgeliefert. Dies erfolgt bei Authenifizierungsverfahren, die dies unterstützen. Der Authenticator sendet zuerst ein "EAP-Success" Paket an den Supplicanten und teilt diesem den Erfolg mit. Anschließend berechnet der Authenticator entsprechende Werte für die Versendung der "EAPOL-Key" Pakete. In drahtgebundenen Netzwerken werden diese Pakete vom Supplicant ignoriert. Die genauen Daten, welche für diese Pakete benötigt werden, findet man im Anhang A.2.2 auf Seite 69.
Die erfolgte Authentifizierung besitzt solange Gültigkeit, bis der Supplicant entweder ein "EAP-Logoff" Paket sendet, um sich abzumelden, oder der Authenticator ein Reauthentifizierung veranlasst. Dieser Mechanismus dient der Erkennung von Supplicanten, welche zum Beispiel durch Hardwareeingriffe vom System getrennt wurden und keine Möglichkeit hatten eine Abmeldung durchzuführen. Während der Reauthentifizierung ist das Port weiterhin aktiv. Die Deaktivierung erfolgt erst nach einer nicht korrekt erfolgten Reauthentifizierung.

3.4 Authentifizierungsverfahren

Zur Authentifizierung eines Nutzers existiert inzwischen eine große Anzahl an verschiedenen Verfahren. So bietet sich die Möglichkeit, für jegliche Art von Netzwerken unter Berücksichtigung ihrer Eigenschaften auch das passende Verfahren zu finden. Die Verwendung dieser Authentifizierungsverfahren wird lediglich durch die Funktionalität des Authenticators und des Authentication-Servers beschränkt. Die meisten Authenticatoren besitzen eine Liste bekannter Verfahren, welche weitergereicht werden können. Alle anderen werden geblockt. Ebenso lassen sich auch nicht alle möglichen Verfahren mittels des Authentication-Servers konfigurieren. Diesbezüglich sollte vor dem Einsatz die Wahl der Soft- und Hardware berücksichtigt werden.
In den folgenden Abschnitten werden kurz die wesentlichen und am weitesten verbreiteten Verfahren vorgestellt. Dabei wird auf ihre Funktionsweise und die Sicherheit, welche durch das Verfahren geboten wird, eingegangen.

3.4.1 Message-Digest 5

Das einfachste und auch schon bei der EAP Version für das Point-to-Point Protokoll festgelegte Verfahren ist die Authentifizierung über Message-Digest 5 (MD5) [22]. Dabei wird der Nutzer über ein Passwort authentifiziert, welches durch einen MD5-Hashwert repräsentiert wird.
Dieses Verfahren besitzt einige durch seine Einfachheit geprägte Probleme. In Netzwerken, bei denen Nutzer Zugang zu Paketen anderer haben, wie zum Beispiel in Funknetzwerken, ist es möglich, die nötigen Informationen zu sammeln und später durch gezieltes Einspielen der Daten eine Authentifizierung zur erlangen. Weiterhin besteht bei diesem Verfahren das Problem, dass auf Grund der schwachen Kryptographie es nicht möglich ist, dynamische Schlüssel zu generieren und anschließend auszutauschen.
Auf Grund dieser Problematik sollte dieses Verfahren nur zu Testzwecken verwendet werden.

3.4.2 Transport Layer Security

Das Verfahren mittels der Authentifizierung über Transport Layer Security (TLS) [23] zählt zu den sichersten Methoden überhaupt. Basis dieser Authentifizierung ist, dass sich Authenticator und Supplicant gegenseitig authentifizieren. Das ermöglicht ein Sicherstellen, dass die Kommunikation nur zwischen den gewünschten Partnern erfolgt.
Die Authentifizierung erfolgt durch den Austausch von X.509 Zertifikaten. Der Ablauf ist wie folgt. Der Supplicant sendet nach der Einigung auf dieses Verfahren ein Paket zur Initiierung des TLS-Sitzungsaufbaus, eine so genannte *ClientHello* Nachricht. Der Authentication-Server antwortet auf diese und sendet eine *ServerHello* Nachricht. Zusätzlich überträgt er sein Zertifikat, welches anschließend vom Supplicant verifiziert wird. Das erfolgt durch eine höhere Instanz, welche als Root Certificate Authority (RootCA) bezeichnet wird. War die Verifizierung erfolgreich, übermittelt der Supplicant dem Server sein eigenes Zertifikat, was der Authentifizierung dient. So wie der

Supplicant prüft der Authentication-Server die Korrektheit des Zertifikates mittels der RootCA. Nach erfolgreicher Verifizierung ist der Supplicant authentifiziert.

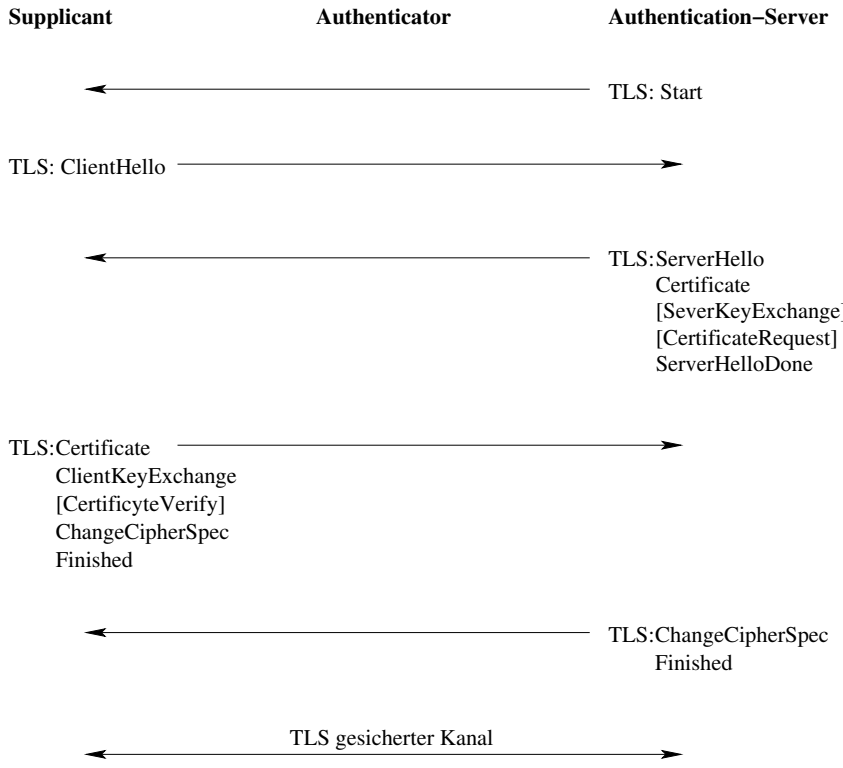

Abbildung 3.5: TLS-Authentifizierung/Verbindungsaufbau

Ein wichtiger Bestandteil dieses Verfahrens ist das Aushandeln eines Geheimnisses, welches als "Master-Secret" bezeichnet wird. Beim Initiieren der TLS-Verbindung werden Zufallswerte übertragen. Der Supplicant bestimmt nach Erhalt des Serverzertifikates ein "Pre-Master-Secret". Dieses verschlüsselt er mit dem öffentlichen Schlüssel des Authentication-Servers, welcher im Zertifikat übermittelt wurde. Der Authentication-Server seinerseits entschlüsselt dies mit seinem privaten Schlüssel. Anschließend besitzen Supplicant und Authentication-Server alle Information, um das "Master-Secret" abzuleiten. Dies ist eine Kombination aus den Zufallswerten und dem "Pre-Master-Secret". Durch die Verschlüsselung des "Pre-Master-Secret" ist es nicht möglich, dass ein Angreifer alle Daten sammeln und nutzen kann. Mit Hilfe des erzeugten "Master-Secrets" können später die Schlüssel für drahtlose Verbindungen abgeleitet werden. Dazu übermittelt der Authentication-Server das "Master-Secret" an den Authenticator. Die Übermittlung ist ebenso durch ein Geheimnis, welches lediglich die Beteiligten kennen, geschützt. Die Sicherheit zur Generierung des "Master-Secrets" beruht auf asymmetrischen Verschlüsselungsverfahren. Diese sind zwar langsamer in der Nutzung, bieten aber die Möglichkeit, Sitzungsschlüssel einfach zu erzeugen und auszutauschen.
Die Sicherheit dieses Verfahrens ist hoch. Es existieren keine bekannten Methoden für Angriffe. Selbst der bei TLS-Verbindungen eingesetzte Man-In-Middle Angriff ist nicht möglich, da das Zertifikat des Authentication-Servers vom Supplicanten verifiziert werden kann. Das einzige Problem,

welches bei der Verwendung dieses Verfahrens entsteht, ist die Verwaltung einer hohen Anzahl an Zertifikaten bei vielen Nutzern. Die Einführung einer so genannten Public Key Infrastruktur erfordert ein hohes Maß an Organisation in größeren Netzwerkstrukturen. Jedoch ist der Einsatz in kleinen Netzwerken vorteilhaft.

3.4.3 Tunneled Transport Layer Security

Bei der Verwendung von Tunneled Transport Layer Security (TTLS) als Authentifizierungsverfahren erfolgt wie bei TLS der Aufbau einer gesicherten Verbindung. Es wird ebenso ein Master-Secret generiert. Mit Hilfe dieses Geheimnisses wird die Verbindung zwischen Supplicant und Authentication-Server verschlüsselt. Dabei kommt ein schnellerer symmetrischer Algorithmus zum Einsatz. Innerhalb von diesem Kanal erfolgt die Übermittlung der Authentifizierungsdaten. Im Gegensatz zu TLS besitzt der Supplicant kein eigenes Zertifikat und kann sich somit nicht beim TLS-Sitzungsaufbau authentifizieren.
Innerhalb des geschützten Kanals können beliebige von den Teilnehmern unterstützte Authentifizierungsverfahren verwendet werden. Gängig sind zum Beispiel Protokolle wie das Password Authentication Protocol (PAP) [24] oder Challenge-Handshake Authentication Protocol (CHAP) [25]. Es können aber auch standardisierte EAP Verfahren wie MD5 oder TLS genutzt werden. Die Verwendung von TLS erhöht jedoch nicht die Sicherheit.
Dieses Verfahren besitzt eine innere und eine äußerer Identität. Die äußere Identität wird als Antwort auf das EAP-Request Identity Paket gesendet. Diese ermöglicht den Aufbau der TLS-Verbindung. Die innere Identität ist die eigentliche des Nutzers, welche für die Authentifizierung benötigt wird. Diese wird ebenso wie weitere Authentifizierungsdaten verschlüsselt übertragen.
Die Sicherheit dieses Verfahren ist nicht so hoch wie bei dem TLS-Verfahren. Da lediglich der Authentication-Server über ein Zertifikat verfügt, kann ein Angreifer sich als dieser ausgeben. Der Supplicant besitzt keine Möglichkeit, den Kommunikationspartner zu überprüfen. Der Aufbau der TLS-Sitzung erfolgt lediglich über die Akzeptanz des Serverzertifikates. Damit ist es möglich, bei dieser Methode durch einen Man-in-the-Middle Angriff nötige Daten zu sammeln und zu dechiffrieren. Anschließend kann der Angreifer diese Daten nutzen, um sich selbst zu authentifizieren.
Ein großer Vorteil bei diesem Verfahren ist der geringe Verwaltungsaufwand. Es wird lediglich ein Zertifikat benötigt. Die Authentifizierung erfolgt durch Verfahren, wo der Nutzer aufgefordert wird, sein entsprechendes Passwort einzugeben, und dies ist in den meisten Infrastrukturen bereits vorhanden. Ebenfalls besteht die Möglichkeit, welche wiederum durch den Aufbau einer TLS-Verbindung ermöglicht wird, dynamische Schlüssel für drahtlose Verbindungen zu erzeugen. Dazu wird das "Master-Secret" an den Authenticator übermittelt.
Dieses Verfahren eignet sich trotz eines gewissen Sicherheitsrisikos zum Einsatz in größeren Netzwerken.

3.4.4 Protected EAP

Protected EAP (PEAP) [26] [27] ist ein von IETF standardisiertes Verfahren und ähnelt sehr dem TTLS-Verfahren. Wie bei TTLS wird zunächst ein sicherer Kanal mittels TLS erzeugt, bevor die eigentlichen Authentifizierung stattfindet. Im inneren Kanal wird anschließend eine neue EAP Sitzung initiiert, bei der existierende Verfahren wie MD5, TLS oder bei Microsoft Windows das Verfahren msCHAP zum Einsatz kommen.
Die Sicherheit dieses Verfahren ist gleich der von TTLS. Es ist ebenfalls möglich, einen Man-in-the-Middle Angriff gegen das Verfahren einzusetzen, um die eigentlichen Daten der Authentifizierung ausfindig zu machen. Die Generierung von dynamischen Schlüsseln wird auf Grund der Etablierung eines TLS-Kanals unterstützt.

3.4.5 Lightweight EAP

Dieses Verfahren wurde durch Cisco Systems definiert. Das ist auch der Grund, warum nur wenige Informationen darüber bekannt sind. Durch eine Protokollanalyse konnten die wesentlichen Protokollmerkmale ausfindig gemacht werden. Die genaue Beschreibung der ermittelten Parameter findet man in [28]. Die Authentifizierung erfolgt über ein Geheimnis, welches der Supplicant und der Authentication-Server kennen. Die Sicherheit dieses Verfahrens ist nicht viel höher als die des MD5-Verfahrens. So besteht als Beispiel die Möglichkeit, einen Wörterbuchangriff einzusetzen. Dieses Problem wurde bereits von Cisco Systems bestätigt [29]. Es besteht jedoch ein Unterschied zum MD5-Verfahren. So ist es möglich, mit Lightweight EAP (LEAP) dynamische Schlüssel für drahtlose Verbindungen zu erzeugen.
Die Verwendung dieses Verfahrens sollte jedoch wie beim MD5-Verfahren nur für Testzwecke eingesetzt werden. Von einem produktiven Einsatz wird auf Grund der bestehenden Sicherheitsrisiken abgeraten.

3.4.6 Weitere Verfahren

Zusätzlich zu den bereits vorgestellten Verfahren existieren noch weitere. Zum Beispiel definiert der Standard des PPP-EAP [21] die Verfahren zur Authentifizierung über One-Time-Passwörter (OTP) und Generic Token Cards (GTC). Weiterhin wurden durch Mobilfunkunternehmen Verfahren zur Authentifizierung mittels "AKA" [30] und "SIM" [31] für den Einsatz in Mobilfunknetzen wie Global System for Mobile Communication (GSM) und Universal Mobile Telecommunications System (UMTS) entwickelt. Durch den einfachen Aufbau des EAP lassen sich auch in Zukunft neue Authentifizierungsverfahren für nahezu alle Situationen entwickeln oder bereits existierende weiter ausbauen.
Im Kapitel 4 auf Seite 24 werden Softwarelösungen für Supplicanten vorgestellt und deren angebotene Verfahren zur Authentifizierung aufgezeigt.

3.5 Sicherheit

Die Sicherheit von 802.1x beruht auf den verwendeten Authentifizierungmethoden. So existieren, wie bereits beschrieben, sichere, teilweise sichere und unsichere Verfahren. Ein kurze zusammenfassende Übersicht ist in Tabelle 3.3 dargestellt. Die dort beschriebenen Angriffsmethoden zielen speziell auf die Authentifizierung. Es sollte aber zusätzlich auf die Sicherheit der eigentlichen Netzwerkinfrastruktur geachtet werden, da auch weitere Methoden für Angriffe in drahtlosen und drahtgebunden Netzwerken existieren.

Verfahren	**Server Authentifizierung**	**Supplicant Authentifizierung**	**Angriffsmöglichkeiten**	**dynamische Schlüssel**
MD5	keine	Passworthash	Man-in-the-Middle, Wörterbuch, Sessionübernahme	nein
TLS	Public Key Zertifikat	Public Key Zertifikat	keine	ja
TTLS	Public Key Zertifikat	PAP, CHAP, EAP Verfahren	Man-in-the-Middle	ja
PEAP	Public Key Zertifikat	EAP Verfahren	Man-in-the-Middle	ja
LEAP	Passworthash	Passworthash	Wörterbuch	ja

Tabelle 3.3: Übersicht über die Authentifizierungsverfahren

Unabhängig von den Angriffen auf die Authentifizierungsverfahren ist es möglich, auch Sitzungen zu übernehmen. Durch gezieltes Senden von EAP Paketen an den Supplicanten und einen anschließenden MAC-Spoofing des Angreifers kann dieser die authentifizierte Sitzung übernehmen und bis zur Reauthentifzierung nutzen. Es sollte daher auf eine angemessene Zeitperiode bis zur erneuten Authentifizierung geachtet werden. Damit wird zwar dieses Problem nicht behoben, aber es wird die Zeit, welche dem Angreifer zur Verfügung steht, verringert.
Wichtig ist noch einmal zu erwähnen, dass durch 802.1x keine Sicherung des Datentransfers nach der Authentifizierung erfolgt. Es ist lediglich ein Authentifizierungsverfahren. In Kabelnetzwerken ist ein Angriff durch verwendete Switchtechnologien meist schwer zu erzielen, aber nicht unmöglich. In Funknetzwerken existieren Verfahren, welche die Daten bei der Funkübertragung sichern (siehe dazu Abschnitt 2.2). Es sollte deswegen auf eine ausreichende Sicherung sensibler Daten bei der Versendung durch das Netzwerk geachtet werden.

Kapitel 4

Authenticatoren, Supplicanten und Authentication-Server

Nach der Verabschiedung des Standards 802.1x Port-Based Network Access Control wurden die notwendigen Voraussetzungen zur Nutzung dieses Authentifzierungsverfahrens in verschiedenen Hard- und Softwareprodukten integriert. Die Nutzung dieser Produkte legt jedoch auch Eigenschaften der Netzwerkumgebung, in der sie eingesetzt werden, fest. Dafür soll in diesem Kapitel ein Überblick über die existierenden Produkte gegeben werden. Es werden Eigenschaften für den Einsatz und ihre Anwendung aufgezeigt.

4.1 Authenticator

Im Bereich der Authenticatoren wird inzwischen eine große Anzahl an Geräten wie Switches oder Access Points angeboten. Zusätzlich zu diesen existieren weitere Produkte, welche durch Software einen Authenticator für bestimmte Umgebungen zur Verfügung stellen. Im folgenden Abschnitt werden Authenticatoren, welche im Zusammenhang mit dieser Arbeit getestet wurden, vorgestellt.

4.1.1 Cisco System - Catalyst 2950

Der von Cisco Systems hergestellte Switch *Catalyst 2950* besitzt einen internen Authenticator. Dieser kann explizit für ein oder mehrere Ports geschalten werden. Der Authenticator unterstützt alle gängigen Verfahren zur Authentifizierung. Mit Hilfe dieses Switches können Umgebungen zur Authentifizierung mittels 802.1x realisiert werden.
Die Verwendung von Switches mit Authenticator in größeren Netzwerken erfordert jedoch ein hohes Maß an Kosten, da die Ports, an welchen sich der Supplicant authentifizieren kann, beschränkt sind. Ein Vorschalten eines Repeaters, um diese Anzahl an Ports zu erhöhen, kann in einzelnen Fällen erfolgen. Diese Eigenschaft ist jedoch stark vom Produkt abhängig. Nicht möglich ist das Weiterleiten der EAPOL Pakete durch den Switch an tiefer liegende Netzwerkinstanzen. Es gibt Geräte die ein Weiterleiten unterstützen, jedoch kann dies nicht als Standard vorausgesetzt werden.

Dadurch ist der Einsatz dieser Geräte für ein 802.1x Struktur mit zentralem Authenticator nicht geeignet. Zusätzlich zu dem hier verwendeten Switch gibt es weitere auch von anderen Herstellern angebotene, welche die Funktionalität eines Authenticators besitzen.

4.1.2 Wireless LAN - Access Points

Ebenso wie bei Kabelnetzwerken existieren Geräte für den Bereich der Funknetzwerke. Diese besitzen ebenfalls ein im Access Point integrierten Authenticator. Im Gegensatz zu Switches haben diese Geräte meist nur geringe Einschränkungen der Zugriffspunkte für die Nutzer. Problematisch wird nur die Teilung des Mediums zur Übertragung der Daten. Je mehr Teilnehmer sich im Funknetzwerk befinden, desto geringer wird die mögliche Transferrate.
Zum Test wurde der von Lucent produzierte Access Point *AP-1000* verwendet. Bei diesem, ist es möglich die Funktionalität von 802.1x zu nutzen. Für ältere Geräte dieser Bauart wurde dazu ein entsprechendes Firmwareupgrade angeboten. Probleme bereitete dieser Access Point bei der Weiterleitung von EAPOL Paketen. Die Pakete wurden jedoch nicht nach der Art und Weise wie beim Cisco Switch gefiltert. Hier ist Filterkriterium der Typ der gesendeten Pakete auf der Verbindungsschicht und nicht die verwendete Multicast-Adresse. Dieses Problem konnten jedoch behoben werden, nachdem eine ältere Firmware eingesetzt wurde, welche das Verfahren nach 802.1x noch nicht beherrscht. Jedoch ist ein Einspielen einer älteren Firmware kritisch.
Allgemein ist zu sagen, dass ältere Hardware, welche zumindest WEP unterstützt, auch die Funktionalität von 802.1x nach einem Firmwareupgrade nutzen kann. Neuerer Hardware, wo Verfahren wie WPA oder nach dem Standard 802.11i zum Einsatz kommen, bieten diese Funktionalität auf jeden Fall, da diese dort für die Authentifizierung benötigt wird.

4.1.3 Software Authenticator - *HostAP* und *MadWifi*

Im Gegensatz zu den Authenticatoren, welche in einem Netzwerkgerät integriert wurden, gibt es Softwarelösungen. Diese Softwarelösungen beschränken sich meist auf Funknetzwerkkarten und werden genutzt, um mit diesen Karten einen Access Point zu realisieren.
Das Softwareprojekt *HostAP* [32] von Jouni Malinen stellt eine solche Softwarelösung dar. Mit Hilfe von Funknetzwerkkarten kann ein Access Point realisiert werden. Dieser besitzt die gleichen Eigenschaften wie ein auf Hardware basierender Access Point. So unterstützt diese Software alle gängigen Funktionen für Funknetzwerke, wie die Verschlüsselung der physischen Schicht mittels WEP oder den Einsatz neuer Technologien wie WPA oder 802.11i. Dafür enthält dieses Produkt einen Authenticator für die Zugangskontrolle mittels 802.1x. Die Software stellt eine vollständige Umgebung mit allen nötigen Treibern zur Verfügung. Jedoch arbeitet diese nur mit Karten zusammen, welche einen Prism 2, 2.5, 5 Chipsatz besitzen.

Eine weitere Open Source Software namens *MadWifi* [33] stellt diese Funktionalität eines Software Access Point für Karten mit einem Atheros-Chipsatz zur Verfügung. Ähnlich wie bei *HostAP* wird dies durch ein Treibermodul für die Karte realisiert.
Probleme bei diesen Produkten sind, dass durch die Verwendung einer Erweiterungskarte für den Computer kein optimaler Standpunkt für den Access Point erreicht werden kann und stets der Einsatz eines Rechners notwendig ist. Somit eignet sich der Einsatz dieser Software nicht für große Netzwerkstrukturen. Für kleine Netzwerkumgebungen kann man jedoch eine preisgünstige und auch gesicherte Lösung realisieren.

4.1.4 Open1x - Authenticator

Ein weiterer Softwareauthenticator ist der als Open Source Projekt entwickelte *Open1x - Authenticator* [36]. Dieses Projekt beschränkt sich jedoch momentan auf die Entwicklung einer Supplicantensoftware.
Der Authenticator wurde für den Einsatz unter Linux entwickelt. Er bietet die Möglichkeit, eine Authentifizierung an einer beim Start definierten Netzwerkschnittstelle durchzuführen. Dieser Authenticator besitzt wenig Funktionalität und kann nur zu Testzwecken genutzt werden. So existiert keine Möglichkeit, mehrere Supplicanten zu verwalten. Eine Blockierung nicht authentifizierter Supplicanten ist ebenfalls nicht realisiert.

4.1.5 Zusammenfassung

Zusammenfassend zu diesem Abschnitt der Soft- beziehungsweise Hardware für Authenticatoren soll erwähnt werden, dass eine Vielzahl zufriedenstellender Authenticatoren existiert. Die Verwendung ist durch die existierende Netzwerkstruktur und Forderung spezifischer Funktionalitäten bestimmt. Für die Verwendung eines zentralen Authenticators gibt es zur Zeit keine Lösung. Es existiert keine Hard- oder Software, welche die Funktionalität besitzt, mehrere Supplicanten an einem Netzwerkinterface zu authentifizieren.

4.2 Supplicanten Software

Durch die Bedeutung der Absicherung von Netzwerken, insbesondere von Funknetzwerken, und der zunehmenden Nutzung von 802.1x sind mehrere Softwarelösungen für Supplicanten entstanden. Dadurch kann mittlerweile jedes Betriebssystem diesen Standard nutzen. Unterschiede zwischen den Supplicanten existieren lediglich in den von ihnen bereitgestellten Authentifizierungsverfahren. In den folgenden Abschnitten werden verschiedene Supplicanten, welche auch im Rahmen dieser Arbeit getestet wurden, vorgestellt und auf die von ihnen angebotenen Verfahren hingewiesen.

4.2.1 Microsoft 802.1x Supplicant

Von Mircrosoft wurde in das Betriebssystem WindowsXP bereits eine Supplicantensoftware integriert. Dieses Software unterstützt jedoch nur einen Bruchteil der möglichen Authentifizierungsverfahren. Es ist möglich, dies durch MD5, PEAP oder TLS zu erzielen. Ein Vorteil ist, dass durch Microsoft eine Schnittstelle zur Verfügung gestellt wird, die es ermöglicht, beliebige Verfahren einfach zu implementieren. Es ist nicht nötig, sämtliche Einzelheiten bei einer Authentifizierung zu beachten. Dafür existieren bereits Funktionen, welche die grundlegende Funktionalität eines Supplicanten gewährleisteten. Somit kann man sich bei der Erweiterung auf das eigentliche Protokoll zur Authentifizierung konzentrieren. Problematisch ist nur, dass die Konfigurationsmöglichkeiten für die internen Funktionen gering sind und dadurch nicht alle Möglichkeiten und nicht alle Szenarien unterstützt werden. So findet sich keine Möglichkeit, die Multicast-Adresse in eine Unicast-Adresse zu ändern, um diesen Supplicanten in durch Switches realisierten Netzwerken einzusetzen.
Ältere Version bleiben bei der Nutzung von 802.1x jedoch außen vor. Lediglich für Windows2000 wurde ein Zusatzprogramm angeboten, welches die Funktionalität bereitstellt und im Zusammenhang mit einem installierten Service Pack 4 genutzt werden kann.
Microsoft bringt auch einige Restriktionen für die Verwendung der Supplicantensoftware in Funknetzwerken mit sich. So ist es zum Beispiel unmöglich, das Authentifizierungsverfahren in nicht verschlüsselten Funknetzwerken zu nutzen. Weiterhin sind auch nur die Verfahren möglich, welche auch nach der Authentifizierung die Möglichkeit besitzen, dynamische Schlüssel zu generieren. Im LAN sind diese Restriktionen nicht vorhanden.

4.2.2 Alfa & Ariss - *SecureW2*

Der von der Firma Alfa & Ariss entwickelte Supplicant stellt eine Erweiterung der Funktionalität des Supplicanten von Windows dar. Wie bereits erwähnt stellt Microsoft mit der Supplicantensoftware auch eine Schnittstelle zur Erweiterung bereit. Basierend auf dieser Schnittstelle wurde von Alfa & Ariss eine Erweiterung vorgenommen, um eine Authentifizierung mittels TTLS zu gewährleisten. Die Erweiterung *SecureW2* [34] bietet die Möglichkeit, verschiedene Verfahren im TLS-Tunnel zu nutzen. So stehen die Standardverfahren wie MD5, TLS und weitere Verfahren wie zum Beispiel PAP zur Verfügung. Der Einsatz von dynamischen Schlüsseln wird durch die Schnittstelle der von Windows zur Verfügung gestellten Funktionen realisiert.
Diese Software ist eine kommerzielle Software. Jedoch erlaubt die Firma Alfa & Ariss die freie Nutzung in nicht kommerziellen Umgebungen.

4.2.3 Wire1x

Um die Lücke zu älteren Versionen von Windows zu schließen, wurde von Wireless Internet Research & Engineering (WIRE) Lab. der Supplicant *Wire1x* [35] implementiert. Dieser basiert auf der von Open1x geschaffenen Struktur. Er bietet die Möglichkeit, sich mittels MD5, TLS, PEAP und

TTLS zu authentifizieren. Die Software unterstützt alle gängigen Windowssysteme und steht zur freien Verfügung. Die Software ist jedoch nicht voll kompatibel zu existierenden Netzwerkkarten für drahtlose Netzewerke. Somit ist es nicht möglich, dynamische Schlüssel zu erzeugen und anschließend zu nutzen. Diese Software nutzt zum Senden der Pakete die "LIBNET" und zum Empfangen die "LIBPCAP". Bei der Verwendung vom WindowsXP mit dem Service Pack 2 wurden die Funktionen für Low-Level Zugriffe geändert. Somit ist es zwingend notwendig, die neusten Versionen dieser Bibliotheken zu installieren. Ein Vorteil dieser Software ist, dass sie unter GNU General Public License (GPL) steht. Dadurch ist ein Erweitern dieser um spezifische Funktionen wie die Verwendung einer Unicast-Adresse möglich. Ein Patch für das Teilprogramm zur Authentifizierung mittels TTLS ist der Arbeit beigefügt.

4.2.4 Open1x - xsupplicant

Der durch Open1x entwickelte Supplicant *xsupplicant* [36], bietet die Möglichkeit, dieses Authentifizierunsverfahren unter Linux zu nutzen. In früheren Versionen war ebenfalls eine Nutzung für MacOS und die BSD-Systeme enthalten. Ältere Codesegmente befinden sich noch im Projekt, wurden jedoch nicht gepflegt, und nach einer Umstrukturierung des Quellcodes ist die Funktionalität nicht mehr gewährleistet.
Dieser Supplicant unterstützt alle gängigen Verfahren zur Authentifizierung wie MD5, TLS, PEAP, LEAP und TTLS. Er bietet die Möglichkeit für eine Authentifizierung im LAN und im WLAN, wobei hier das Erzeugen dynamischer Schlüssel möglich ist. Diese Schlüssel werden mit Hilfe der *Linux Wireless Extension* und den *Wireless Tools* [37] einem Open Source Projekt von Jean Tourrilhes, als Parameter der Netzwerkkarte übergeben. Jedoch ist eine Verwendung von WPA nicht garantiert. Dieser Teil befindet sich noch im Entwicklungsstatus. Ebenfalls ist eine Nutzung dieser Software in unverschlüsselten Funknetzwerken möglich.
Als Besonderheit bietet die Software die Einstellung einer Unicast-Adresse für die Pakete der Verbindungsschicht. Dadurch kann eine Anwendung in Netzwerken, welche durch Switches realisiert sind, erfolgen.

4.2.5 Jouni Malinen - wpa_supplicant

Ebenso wie der Supplicant des Open1x Projektes stellt der *wpa_supplicant* [38] die Funktionalität von 802.1x für Linux zur Verfügung. Der Unterschied zwischen diesen beiden Projekten ist, dass der *wpa_supplicant* nicht nur eine Unterstützung für WEP bietet, sondern ebenso für WPA und 802.11i. Dieser bietet für die Verwendung ein vollständiges Schlüsselmanagment an. Jedoch ist die Funktionalität auf bestimmte Chipsätze beziehungsweise Treiber beschränkt, da für das Setzen der Schlüssel direkt mit dem Treiber gearbeitet wird.
Dieser Supplicant kennt alle üblichen Verfahren wie TLS, TTLS, PEAP und LEAP zur Authentifizierung. Die Verwendung von MD5 ist auf Grund der fehlenden Eigenschaft zur Schlüsselgenerierung nicht implementiert, soll aber in einer späteren Release hinzugefügt werden.

4.2.6 MacOS X Panther 802.1x Supplicant

Für das Betriebssystem MacOS X Panther bietet die Firma Apple Computer Inc. einen 802.1x [39] Supplicanten an. Dieser unterstützt zwar die üblichen Verfahren zur Authentifizierung, arbeitet jedoch nur mit einer beschränkten Anzahl von Funknetzwerkkarten zusammen. Die Verwendung in drahtgebundenen Netzwerken wird nicht ermöglicht.
Die Nutzung dieses Supplicanten beschränkt sich auf Eigenheiten und Eigenschaften von MacOS. Dadurch ist es nur bedingt möglich, diesen einzusetzen.

4.2.7 Kommerzielle Supplicanten

Zusätzlich zu den vorgestellten Supplicanten gibt es eine Reihe von Supplicanten aus dem kommerziellen Bereich. Der *Odyssey* [40] Supplicant der Firma Funk Software Inc. ist für die Nutzung in Windowssystemen sowie auf Pocket PC's verfügbar. Er stellt alle sicheren Verfahren für die Authentifizierung zur Verfügung. Ein weiterer Supplicant der Firma Meetinghouse Data Communications mit dem Namen *AEGIS* [41] gehört ebenfalls zur Kategorie der kommerziellen Supplicanten. Auch dieser besitzt alle üblichen Verfahren zur Authentifizierung und kann in fast jedem Betriebssystem eingesetzt werden.

4.2.8 Zusammenfassung

Durch die hohe Vielzahl an freien und kommerziellen Supplicanten ist der Einsatz dieses Authentifizierungsmechanismusses in nahezu jeder Netzwerkumgebung möglich. Abschließend soll die Tabelle 4.1 eine Übersicht über die vorgestellten Supplicanten geben. Dabei werden ihre wesentlichen Eigenschaften und Einsatzgebiete gegenübergestellt.

Name	**Hersteller**	**Verfahren**	**Systeme**	**dyn. Schlüssel WEP/WPA, 802.11i**
MS 802.1x Supplicant	Microsoft Inc.	MD5(LAN), TLS, PEAP	WindowsXP	ja/ja
SecureW2	Alfa & Ariss	TTLS	WindowsXP	ja/ja
Wire1x	Wireless Internet Research & Engineering (WIRE) Lab.	MD5, TLS, TTLS, PEAP	Windows XP/ NT/ 2k/ 98/ ME	nein/nein
xsupplicant	Open1x	MD5, TLS, TTLS, PEAP, LEAP	Linux	ja/nein
wpa_supplicant	Jouni Malinen	TLS, TTLS, PEAP, LEAP	Linux	ja/ja
Mac 802.1x Supplicant	Apple	MD5, TLS, TTLS, PEAP, LEAP	MacOS X Panther 10.3.x	ja (Aironet Karten)/ ?
AEGIS	Meetinghouse Data Communications	MD5, TLS, TTLS, LEAP, PEAP	Windows XP/ NT/ 2k/ 98/ ME, Pocket PC, MacOS, Palm, Linux, Solaris	ja/ja
Odyssey	Funk Software Inc.	TLS, TTLS, PEAP, LEAP	Windows XP/ 2k/ 98/ ME, Pocket PC, Windows Mobile	ja/ja

Tabelle 4.1: Übersicht über die existierende Supplicantensoftware

4.3 Authentication-Server

Als Authentication-Server werden Remote Authentication Dial-In User Service (RADIUS) Server eingesetzt. Diese besitzen Verfahren für Authentifizierungen, Authorisierungen und Abrechnung von genutzten Ressourcen. Diese Server werden in vielen Szenarien verwendet. Zur Kommunikation dient ein spezifiziertes Protokoll [42]. Dieses Protokoll wird zum Austausch der Daten zwischen Authenticator und dem Authentication-Server genutzt. Dafür bietet dieses Protokoll einen entsprechenden Mechanismus, die Daten, welche der Supplicant übermittelt, zu übertragen.
Die "FreeRADIUS" Software [43, 44] ist eine Open Source Software für Linux und stellt einen vollständigen RADIUS Server zur Verfügung. Dieses Produkt wird als vorkompilierte Software in verschiedenen Linux Distributionen angeboten. Jedoch sind diese meist nicht auf dem neusten Stand. Die aktuellste Version findet man im Internet unter *www.freeradius.org*.
Bei dieser Arbeit wurde zuerst die Version 0.9.3 verwendet. Diese besitzt schon die Funktionalität zur Authentifizierung von Nutzern durch den Standard 802.1x. Dafür ist ein spezielles Modul vorhanden, welches diese Anfragen bearbeitet. In dieser Version werden aber noch nicht alle Verfahren zur Authentifizierung angeboten. So existieren lediglich die Verfahren MD5, TLS, OTP und LEAP. Während der Testphase des RADIUS Server wurde eine neue Release des Servers freigeben. In der Version 1.0 wurden speziell die Verfahren für 802.1x erweitert. So wird jetzt auch die Authentifizierung mittels TTLS, PEAP und GTC angeboten. Diese Version stellt erstmals alle gängigen Verfahren zur Verfügung. Die Konfiguration des Servers ist relativ leicht durchzuführen. Im Anhang D auf Seite 88 befindet sich eine Beispielkonfiguration für diesen RADIUS Server.
Zusätzlich zu dem getesteten, existieren weitere RADIUS Server für andere Betriebssysteme außer Linux. Diese Server bieten die gleiche oder sogar eine bessere Funktionalität als der "FreeRADIUS" Server. Die Auswahl des RADIUS Servers ist stark von der bereits existierenden Netzwerkumgebung und den darin befindlichen Ressourcen abhängig. Diese weiteren Server wurden jedoch im Rahmen dieser Arbeit nicht getestet.

Kapitel 5

802.1x Unterstützung an der Technischen Universität Chemnitz

Dieses Kapitel beschäftigt sich mit dem Hauptziel der Arbeit. Es soll festgestellt werden, ob ein Einsatz des durch den Standard 802.1x definierten Authentifizierungsverfahrens an der Technischen Universität Chemnitz möglich ist und realisiert werden kann. Dazu wird im ersten Abschnitt dieses Kapitels die momentane Architektur des Rechnernetzwerkes betrachtet und existierende Verfahren analysiert, um eine parallele Nutzung von 802.1x zu ermöglichen. Aufbauend auf diese Analyse wird die mögliche Umsetzung von 802.1x erörtert. Dafür werden nötige Voraussetzungen und Entscheidungen getroffen sowie mögliche Probleme aufgezeigt und falls möglich gelöst.

5.1 Analyse der existierenden Umgebung

Die Technische Universität Chemnitz bietet momentan die Möglichkeit an, dass authorisierte Nutzer sich mit eigenen Geräten mit dem Rechnernetzwerk verbinden können, um damit auf interne sowie externe Ressourcen zu zugreifen. Die eingesetzten Systeme unterscheiden sich nicht nur im Einsatzgebiet sondern auch in ihrer Funktionsweise. Es stehen Verfahren für das Funk- und das Kabelnetzwerk zur Verfügung. Um eine Integration des 802.1x Standard zu realisieren, müssen diese Verfahren zuerst betrachtet werden, da sie weiterhin parallel zur Verfügung stehen sollen.
Die folgenden Abschnitte zeigen den aktuellen Zustand dieser Systeme. Abhängig vom eingesetzten Netzwerkaufbau und ihrer Funktion beziehungsweise Arbeitsweise werden sie kurz analysiert.

5.1.1 LAN Umgebung

Das Kabelnetz der TU Chemnitz erstreckt sich über mehrere Gebäude an verschiedenen Standorten. In den Räumen sind entsprechend der Nutzung mehrere Schnittstellen vorhanden und mit dem Netzwerk verbunden. Nicht alle Schnittstellen sind für den Einsatz von Geräten des Rechenzentrums nötig. Es existieren zum Beispiel freie Dosen in Vorlesungsräumen. Die Verbindung zwischen den Dosen und dem Rechnernetzwerk erfolgt über mehrere Switches. Um eine Trennung zum internen

Netzwerk zu realisieren, werden Virtual LAN's (VLAN) [45] genutzt. Diese besitzen die Möglichkeit, Ports beziehungsweise Dosen an unterschiedlichen Stellen und in unterschiedlichen Teilen des durch Switchtechnologien getrennten Netzwerkes zu einem physischen Subnetz zu verbinden.
Mit Hilfe dieser Technologie, welche durch die verwendeten Switches hardwaremäßig unterstützt wird, kann eine Kapselung des freien Netzwerkes vom intern genutzten Rechnernetzwerk erreicht werden. Dadurch wird nicht nur eine Sicherung des Rechnernetzwerkes erreicht, sondern es kann auch an zentraler Stelle ein Rechner zur Authentifizierung und Zugangssteuerung eingesetzt werden.

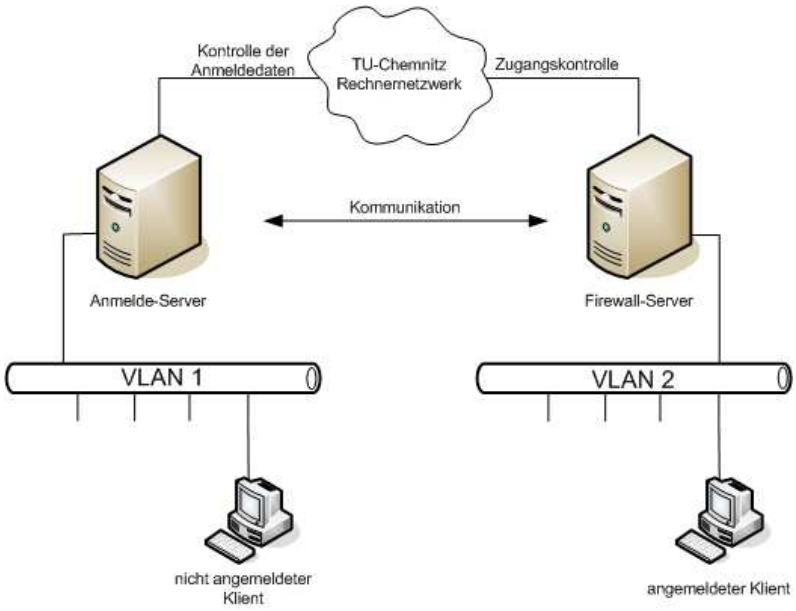

Abbildung 5.1: Aufbau LAN-Zugangssystem

Das momentan genutzte Verfahren des Portmanager-Systems [46] wurde von André Breiler entwickelt. Es basiert auf der Verwendung verschiedener VLAN's. Die Abbildung 5.1 zeigt der Aufbau dieser Lösung. Alle nicht durch eine Authentifizierung freigeschaltenen Dosen befinden sich in dem "VLAN 1", dem so genannten "Anmelde-Netzwerk". In diesem Netz sorgen verschiedene Programme dafür, dass nur eine Kommunikation innerhalb von diesem möglich ist. So können lediglich Verbindungen zwischen dem Gerät des Nutzers und dem "Anmelde-Server" aufgebaut werden. Die geforderte Anmeldung erfolgt über ein Webformular. Dort besteht die Möglichkeit, spezifische Daten zur Anmeldung anzugeben. Im Hintergrund, das heißt, nach dem Abschicken der Formulardaten, prüft ein CGI-Skript die Korrektheit dieser und beginnt bei Erfolg mit der Freischaltung des Nutzers. Dies wird durch ein mehrstufiges Verfahren erreicht. Zuerst erhält der Nutzer eine Erfolgsmeldung vom Webserver des "Anmelde-Netzwerkes". Anschließend wird über das Simple Network Management Protcol (SNMP) das Port des Switches, mit welchem sich der Nutzer verbunden hat, ermittelt. Ist dieses Ermittlung erfolgreich, wird dieses Port entsprechend der Nutzerdaten in das andere "VLAN 2" umgeschalten. In den meisten Fällen erfolgt eine Umschaltung in das "Firewall-Netzwerk", welches in der Abbildung 5.1 bereits ersichtlich ist. Die Webseite, welche den Erfolg der Authentifizierung zeigt, wird durch einen Refreshmechanismus nach kurzer Zeit erneut gela-

den. Wird dieser Mechanismus in Gang gesetzt und die Umschaltung des Ports war erfolgreich, erhält der Nutzer eine anschließende Erfolgsmeldung. Sind Probleme aufgetreten, befindet sich der Nutzer weiter im "Anmelde-Netzwerk" und erhält eine entsprechende Fehlermeldung. Bei Erfolg steht dem Nutzer der Zugang zum Rechnernetzwerk zur Verfügung. Die Kontrolle, ob das Gerät des Nutzers weiterhin verfügbar ist, wird durch ein zusätzliches Programm ermittelt. Dieses sendet ICMP-Ping Pakete und wartet auf die entsprechende Antwort. Erfolgt diese Antwort nicht, werden alle durchgeführten Maßnahmen zurückgesetzt und das Port in das "Anmelde-Netzwerk" zurück geschalten.
Ein Problem bei diesem Zugangssystem ist, dass der Zeitaufwand für das Umschalten des Ports hoch ist. Dies könnte durch ein verbessertes Suchverfahren für das umzuschaltende Port für den Nutzer freundlicher gestaltet werden. Weiterhin können Probleme auftreten, weil viele Prozesse synchronisiert werden müssen. Ebenso ist es möglich, dass Ports auf Grund von Fehlschlägen bei der Suche oder durch fehlende Informationen nicht umgeschalten werden können.
Für eine ausführliche Beschreibung des Portmanager-System wird auf die Diplomarbeit *Differenzierte Bereitstellung von Internetdiensten in öffentlichen Bereichen der Universität* [46] von André Breiler hingewiesen.

5.1.2 WLAN Umgebung

In den letzten Jahren wurde zum existierenden Kabelnetzwerk ein Funknetzwerk an der TU-Chemnitz [47] eingerichtet und stetig erweitert. Diese Art von Netzwerken ist flexibler einsetzbar als Kabelnetzwerke. Jedoch stellen diese auch ein erhöhtes Sicherheitsrisiko dar. Die Einführung einer Zugangsmanagementlösung war somit zwingend erforderlich. Dieses Verfahren wurde weit vor der Festlegung der Verfahren WPA und 802.11i entwickelt. Der einzig mögliche Schutz, war der Einsatz von WEP. WEP ist zwar konfiguriert, wird aber nicht standardmäßig genutzt, da dieser keinen ausreichenden Authentifizierungsmechanismus besitzt und die Sicherheit nicht viel höher ist als bei unverschlüsselten Netzwerken, wie der Abschnitt 2.2 bereits zeigte. Aus diesem Grund und wegen des freien Einsatzes wird das Funknetzwerk der TU-Chemnitz vorwiegend unverschlüsselt betrieben.
Das WLAN besteht ebenfalls aus einer Vielzahl an Access Points in den unterschiedlichsten Bereichen der Universität. Diese sind über Kabel mit dem Rechnernetzwerk verbunden. Ebenso wie beim Kabelnetzwerk befinden sich diese Access Points in einem gesonderten VLAN. Dadurch ist der Zugriff nur auf Rechner innerhalb dieses VLAN's möglich. Für den Zugang zu anderen Netzwerkteilen wurde ein zentraler Authentifizierungsserver eingesetzt, welcher nach erfolgreicher Authentifizierung eines Nutzers den Zugang auf interne und externe Ressourcen freigibt und kontrolliert.
Dieses System für die Zugangssteuerung im WLAN wurde durch Mirko Parthey [48] entwickelt. Im Gegensatz zu dem im LAN-Bereich eingesetzten Verfahren wird lediglich ein VLAN verwendet, da die angemeldeten Nutzer am Access Point nicht durch SNMP umgeschaltet werden können. Ansonsten ähnelt das Prinzip der Anmeldung dem des Portmangers. Zuerst erhält der sich an-

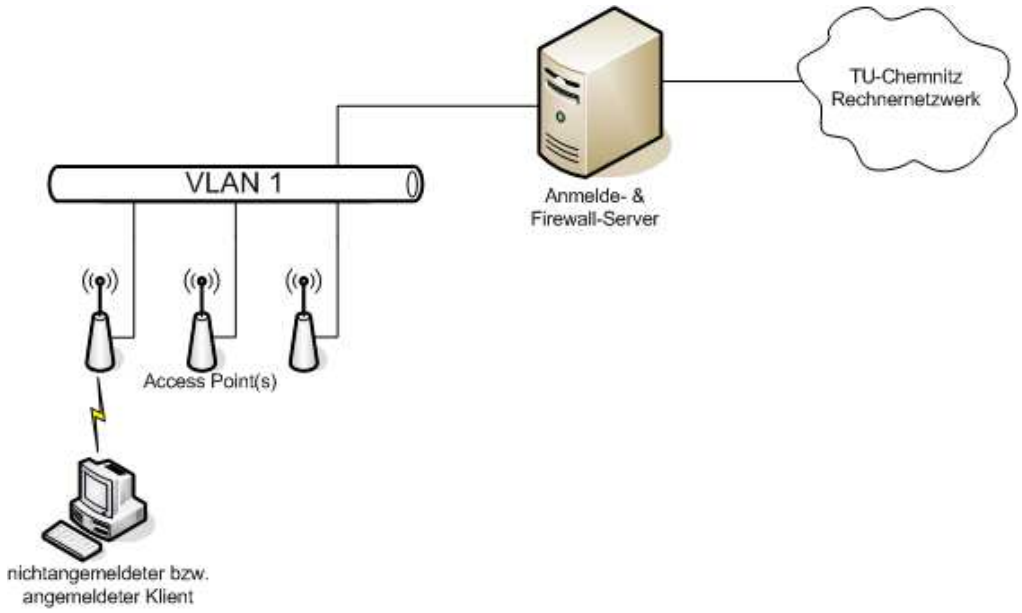

Abbildung 5.2: Aufbau WLAN-Zugangssystem

meldende Nutzer eine vom DHCP zugewiesene IP-Adresse und gegebenfalls Informationen über DNS und Gateway. Anschließend erfolgt die Anmeldung über ein Webinterface. Ein im Hintergrund ausgeführtes CGI-Skript auf dem Webserver kontrolliert diese Anmeldedaten und schaltet nach erfolgreicher Anmeldung die IP-Adresse des Nutzers frei. Dies erfolgt durch das spezifische Setzen verschiedener Regeln im IP-Filter des zentralen Rechners. Die Prüfung, ob der Nutzer sich noch im Netzwerk befindet, wird wie im LAN durch das Senden von ICMP-Ping Paketen erreicht. Erfolgt keine Antwort, werden alle Informationen, welche bei der Authtenifizierung gespeichert beziehungsweise in den verschiedenen Diensten eingetragen wurden, zurückgesetzt.
Die ausführliche Beschreibung dieses Systems kann in der Diplomarbeit *Zugangsmanagement für Wireless LAN* [48] von Mirko Parthey nachgelesen werden.
Zusätzlich zu diesem Verfahren bietet das Rechenzentrum die Nutzung eines Virtual Privat Network an. Dieses wird mit Hilfe der IPSec Technologie realisiert und kann im WLAN und für den Zugang aus dem Internet genutzt werden. Weiter Informationen zur Nutzung des VPN findet man unter [47].

5.1.3 Authorisierte Nutzer

An der Universität sind viele Nutzer berechtigt, die Computer in den entsprechenden Räumen zu nutzen. Doch nicht jeder ist gleichzeitig auch für die Nutzung der freien Zugänge berechtigt. Zu der Kategorie der authorisierten Nutzer gehören Mitarbeiter und Studenten, welche mit Erfolg die Prüfung für das *Zertifikat der Internetnutzung (ZIN)* [49] abgeschlossen haben. Nur diese Nutzer besitzen die Erlaubnis, die öffentlichen Schnittstellen zu verwenden. Diese Einschränkung sollte beim Einsatz von 802.1x ebenso berücksichtigt werden. Zusätzlich bieten die existierenden Verfah-

ren für LAN und WLAN die Möglichkeit eines Gastzuganges an. Dieser beschränkt die Nutzung des Netzwerkes auf lokale Ressourcen. Dazu zählen universitätsintere Webseiten. Ein Zugriff auf Ressourcen außerhalb des Universitätsnetzwerkes ist mit diesem Account nicht möglich.

5.2 Voraussetzung für eine Integration von 802.1x

Es gibt einige wichtige Punkte, welche als Voraussetzung für die Integration gelten. Diese gliedern sich in den Aufbau des Netzwerkes und in die benötigten Servertechnologien. Die folgenden beiden Abschnitte fassen diese Voraussetzungen zusammen.

5.2.1 Aufbau der 802.1x Umgebung

Um die weitere Nutzung der bestehenden Verfahren zu gewährleisten, erfolgt ein Aufbau wie im LAN und WLAN. Es kommt ein zentraler Authenticator zum Einsatz, welcher auf dem bereits existierenden "Anmelde-Server" installiert werden muss. Dadurch ist gesichert, dass alle Verbindungen über diesen abgearbeitet werden und dadurch ein zentral gesteuerter Sicherungsmechnismus wirksam werden kann. Im Gegensatz zum Portmanager im LAN sollte jedoch nur ein VLAN genutzt werden, da ein Umschalten die Folgekommunikation beeinträchtigt. Der Grund dieses Problems liegt in der Anwendung des Reauthentifizierungsmechanismus. Dieser prüft die Existenz und die Authorität des Nutzers nach einer bestimmten Zeit neu. Dies erfolgt über die gleiche Schnittstelle wie die erste Authentifizierung. Daraus folgt, dass durch ein Umschalten des VLAN's dieser Mechanismus schwer nutzbar ist. Die Abbildung 5.3 zeigt den optimalen schematischen Aufbau einer 802.1x Umgebung mit zentralem Authenticator.

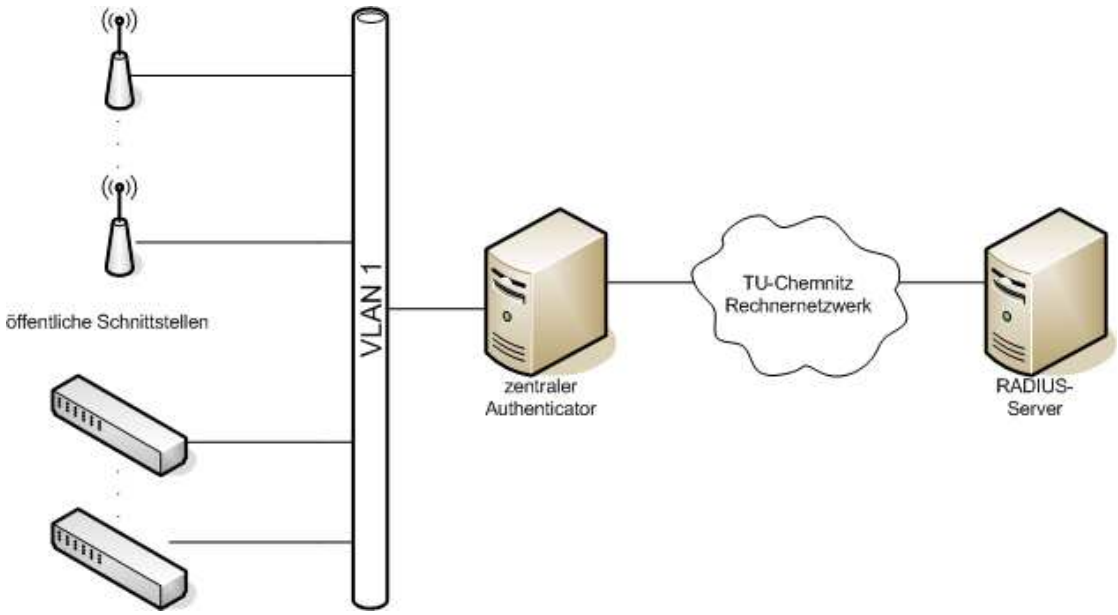

Abbildung 5.3: 802.1x Umgebung mit zentralem Authenticator

Die Abbildung zeigt, dass alle freien Zugriffspunkte sich in einem VLAN befinden. Dadurch folgt, wie bereits erwähnt, eine Trennung zum internen Teil des Rechnernetzwerkes. Außerdem werden alle Nutzer, welche sich mit dem öffentlichen Teil verbinden, über den Authenticator authentifiziert und der Zugang durch diesen überwacht.

5.2.2 Benötigte Soft- und Hardware

Zusätzlich zum benötigten Authenticator sind weitere Softwarelösungen notwendig. Wichtig für die Authentifizierung ist ein Remote Access Dial-In User Service (RADIUS) Server, welcher die Authentifizierungdaten prüft. Dieser sollte, wie in Abbildung 5.3 zeigt, direkt mit dem Authenticator verbunden sein, da eine Kommunikation zu jeder Zeit möglich sein muss. Zu beachten ist, dass der eingesetzte RADIUS Server die verwendeten Authentifizierungsverfahren, welche in der Umgebung zum Einsatz kommen sollen, verarbeiten kann. Zusätzlich wird ein Dynamic Host Configuration Protocol (DHCP) Server benötigt. Dieser liefert an den Nutzer für die spätere Nutzung die entsprechenden Information über IP-Adresse, Nameserver und Gateway. Dieser DHCP Server kann auf dem Rechner, wo bereits der Authenticator läuft, installiert werden. Als letztes benötigt dieser zentrale Rechner die Möglichkeit, einen Sperrmechanismus zu realisieren, um entsprechend der Authentifzierung Verbindungen in das interne Netz zuzulassen oder zu sperren. Dazu können zum Beispiel die unter Linux verfügbaren IP-Tables verwendet werden. Diese bieten die Möglichkeit, entsprechende Sperrmechanismen an den unterschiedlichen Protokollebenen zu realisieren.
Als Hardwarevoraussetzung ist lediglich ein einzelner Rechner für den Authenticator notwendig. Auf diesem können alle benötigten Softwarelösungen installiert werden. Natürlich ist eine Verteilung einzelner Serverprogramme ebenso möglich. Die Belastung des Rechners, um seine Leistungsfähigkeit abzuschätzen, hängt von der Anzahl der Ports ab, an welchen sich Nutzer authentifizieren können. Außerdem wird diese Leistungsfähigkeit von der Konfiguration des Authenticators und des verwendeten Authentifizierungsverfahrens bestimmt. Der Rechner muss dazu mindestens über zwei Netzwerkschnittstellen verfügen. Eine dient dazu den Rechner mit dem öffentlichen Netzwerk zu verbinden und eine weitere ist für den Zugriff auf die interne Netzwerkstruktur zuständig. Diese kann dann auch für die Kommunikation zum RADIUS Server verwendet werden.

5.3 Umsetzung der 802.1x Technologie

Für die Umsetzung einer 802.1x Umgebung wurden einige Parameter im Vorfeld festgelegt. So sollte die Rechner, auf welchem der Authenticator installiert wird, das Betriebssystem Linux nutzen. Dieses sollte mindestens eine Kernelversion 2.4.x besitzen. Weiterhin ist die Bereitstellung der IP-Table Funktionalität notwendig. Unter diesen Parametern wurde die nötige Umsetzung realisiert. Die folgenden Abschnitte betrachten diese Umsetzung detaillierter und beschreiben die Maßnahmen näher, welche für die einzelnen Dienste getroffen wurden.

5.3.1 Authenticator

Da für die Rolle eines zentralen Authenticator für diese Umgebung momentan keine Software existiert, musste eine andere Lösung gefunden werden. Ein einziges Projekt, welches hätte adaptiert werden können, ist der Authenticator des Open1x Projektes. Dieser besitzt eine nahezu vollständige 802.1x Implementierung. Das Problem dieser Software ist jedoch, dass nötige Strukturen für den Einsatz fehlen und eine Integration dieser nicht auf Grund dort getroffener Implementierungsentscheidungen realisierbar ist. Dazu kommt, dass diese Software lediglich zur Testzwecken entwickelt wurde und dadurch die Implementierung Fehler und keine günstige Modularisierung aufweist. Benötigte Eigenschaften, welche dieser Software fehlen, sind die Möglichkeit, mehrere Nutzer an einem Interface zu authentifizieren, sowie die Bereitstellung eines Systems für den Sperrmechanismus. Zusätzlich fehlen einige nicht für Testzwecke benötige Funktionen der 802.1x Funktionalität.
Diese Eigenschaften hätten eine fast vollständige Umstrukturierung der Software gefordert. Aus diesem Grund wurde eine eigenständige Implementierung eines Software-Authenticators durchgeführt, welche für den Einsatz im Rechnernetzwerk der TU Chemnitz alle Voraussetzungen erfüllt. Eine detaillierte Beschreibung dieser Software erfolgt im nächsten Kapitel.

5.3.2 Remote Access Dial-In User Service Server

Der für die Authentifizierung notwendige RADIUS Server existiert bereits an der TU Chemnitz. Dieser wird momentan zur Authentifizerung von Nutzern für unterschiedliche Dienste verwendet. Wichtig für den Einsatz in einer 802.1x Umgebung ist die Funktionalität für eine durch EAP stattfindende Authentifizierung. Dazu ist es notwendig, dass das Verfahren TTLS und eventuell PEAP bereitgestellt wird. Diese Verfahren gelten als nahezu sicher und bieten die Möglichkeit, Nutzer mit ihren Nutzernamen und ihrem Passwort zu authentifizieren. Die wesentliche Eigenschaft dieser Verfahren ist die geschützte Übermittlung der Nutzerdaten.
Für den Einsatz wird der FreeRADIUS Server vorgesehen. Dieser sollte mindestens in der Version 1.0 verfügbar sein, da erst diese Version die benötigten Verfahren unterstützt. Für den Einsatz dieser Verfahren ist es notwendig, ein Serverzertifikat, welches zum Aufbau der TLS-Verbindung benötigt wird, zu erstellen. Dieses muss ebenso durch eine Certification Authority zertifiziert sein. Weiterhin müssen entsprechende Informationen über die authorisierten Nutzer verfügbar sein. Diese sind bereits in zentralen Datenbanken des Rechenzentrums gespeichert. Lediglich ein Zugriff auf diese Datenbanken muss dem RADIUS Server möglich sein. Da der RADIUS Server viele Verfahren zur Authentifizierung unterstützt und jedes die Daten anders verschlüsselt beziehungsweise sichert, sollte als inneres Protokoll bei der Authentifizierung PAP verwendet werden. Dort besteht die Möglichkeit, die Nutzerdaten unverschlüsselt zu übertragen, um dem RADIUS Server die entsprechende Wandlung in die einzelnen Verfahren zu überlassen. Dies ist zwar keine vollkommen sichere aber eine sehr flexible Variante und ein Angriff ist durch die Nutzung einer TLS-Verbindung nahezu unmöglich. Lediglich der RADIUS Server kann die Daten entsprechend entschlüsseln. Der Authenticator besitzt beim Ablauf der Authentifizierung keine Möglichkeit, eine Kenntnis der Da-

ten zu erhalten. Unabhängig von dieser Kombination stehen weitere Verfahren zur Verfügung. Die Verwendung muss jedoch durch die Verfügbarkeit des Supplicanten, des RADIUS Servers, des Authenticators sowie der externen Datenbanken gewährleistet sein. Eine Erweiterung von EAP um ein eigenständiges Verfahren, beziehungsweise das Hinzufügen eines Moduls für den RADIUS Server, wäre ebenso denkbar.
Im Anhang D befindet sich ein Auszug der Konfiguration des RADIUS Servers, welcher diese Verfahren mit den beschriebenen Parametern besitzt.

5.3.3 Dynamic Host Configuration Protocol Server

Eine weitere Technologie, welche eingesetzt wird, ist ein Dynamic Host Configuration Protcol (DHCP) Server. Dieser besitzt die Möglichkeit, die Geräte der Nutzer dynamisch zu konfigurieren. Er liefert Angaben über die zu verwendende IP-Adresse, den Nameserver und Gateways. Von einer statischen Konfiguration der Nutzergeräte wird abgeraten, da dies durch die Fluktuation der Nutzer einen hohen Verwaltungsaufwand hervorruft. Außerdem steht die Verwendung statischer Konfigurationen im Gegensatz zu dem Prinzip der öffentlichen Schnittstelle. Der Einsatz dieser Technologie zur Konfiguration wird von den meisten Betriebssystemen unterstützt. Dadurch ist keine zusätzliche Software auf der Nutzerseite erforderlich.
Für den Einsatz in der 802.1x Umgebung kann auf existierende DHCP Server zurückgegriffen werden. Da dieses Verfahren parallel zum WLAN- und LAN-Zugang integriert wird, können die bereits dort verwendeten Server zu Einsatz kommen. Die Verfahren im LAN und WLAN beeinflussen zwar die DHCP Konfiguration, kommen aber nicht mit dem 802.1x Verfahren in Konflikt. Dies hat jedoch einige Einschränkungen zur Folge. So ist es nicht möglich, nach erfolgter Anmeldung die IP-Adresse des privaten Adressbereichs, welcher standardmäßig genutzt wird, in eine öffentliche IP-Adresse zu ändern. Dies würde seitens des Authenticators ebenso eine Beeinflussung der DHCP Konfiguration erfordern. Dadurch könnten Konflikte zwischen den parallel angebotenen Verfahren entstehen. Sollte der Authenticator in einer eigenen Umgebung installiert werden, könnte diesbezüglich eine Erweiterung vorgenommen werden.

5.3.4 Sperrmechanismus

Zum Schutz des internen Netzwerkes sollte bei dieser zentralen Lösung ein Sperrmechanismus für nicht authentifizierte Nutzer eingesetzt werden. Dieser kann sich auf einen einfachen Filter beschränken. Eine Möglichkeit, dies zu erreichen, kann, wie in der Voraussetzung erwähnt, mit Hilfe der unter Linux existierenden "IP-Tables" erfolgen. Der Aufbau und das Zusammenwirken wird in Abbildung 5.4 gezeigt. Die detaillierte Funktionsweise dieser Technologie soll in diesem Dokument nicht näher betrachtet werden. Eine ausgiebige Dokumentation findet man unter [50]. Folgend werden lediglich Eigenschaften für die Nutzung in der zu realisierenden 802.1x Umgebung vorgestellt. Eine besondere Eigenschaft dieser Technologie ist, dass Pakete nicht nur auf IP-Ebene sondern auch auf der Verbindungsschicht gefiltert werden können. Die Funktionalität, welche dort zur Verfügung

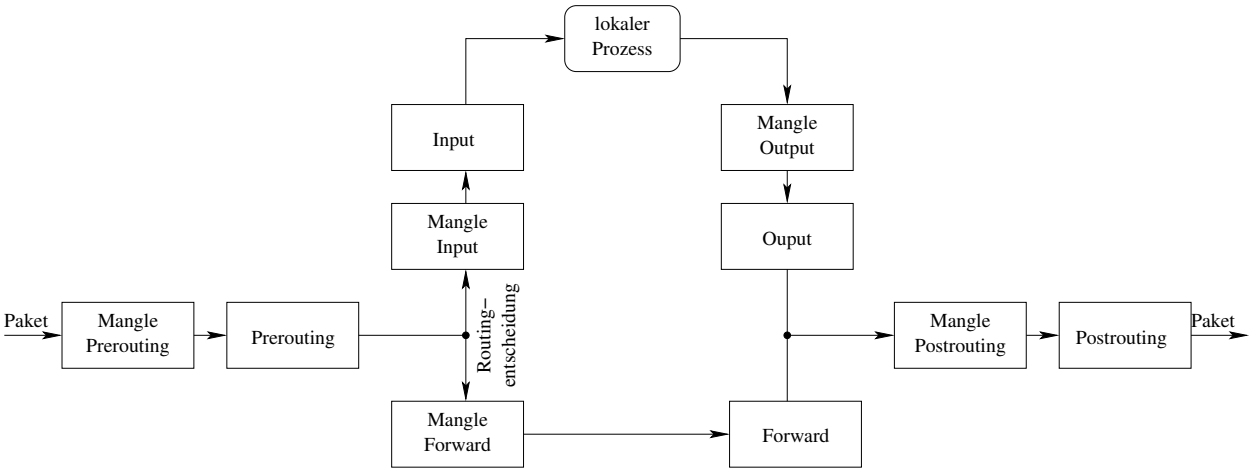

Abbildung 5.4: Aufbau der IP-Tables

steht, ist jedoch stark beschränkt. Es ist aber möglich, Pakete bestimmte Sender, das heißt, mit einer bestimmten Source-MAC-Adresse zu filtern. Dies ist zwingend erforderlich, da zwischen Authenticator und Supplicant nur die MAC-Adressen zur Verfügung stehen. Eine weitere Eigenschaft ist, dass die Pakete in der Filterkette **Mangle-Prerouting** markiert und somit klassifiziert werden können. Diese Markierung kann in den folgenden Instanzen ausgewertet und daraufhin Filterentscheidungen getroffen werden. Zusätzlich dazu muss ebenso ein Weiterleiten von Paketen authentifizierter Nutzer erfolgen. Dies wird über einen Eintrag in der Filterkette **Postrouting** erreicht. Dort wird eine Network Address Translation (NAT) durchgeführt. Dadurch folgt, dass in allen Paketen, welche den Rechner über das Interface zum internen Netzwerk verlassen und auch die Berechtigung dafür besitzen, die Sender IP-Adresse in die des zentralen "Anmelde-Servers" umgeschrieben wird. Die "IP-Tables" Implementation speichert diesen Vorgang und leitet entsprechende Pakete, welche als Antworten auf die gesendeten Pakete eintreffen, an den eigentlichen Sender zurück. Diese drei Eigenschaften sind für den Software-Authenticator nötig. Ein einziges Problem, welches nicht lösbar ist, ist das Filtern beziehungsweise Markieren von Paketen in Abhängigkeit einer Destination-MAC-Adresse. Somit besteht mit den "IP-Tables" keine Möglichkeit, die Kommunikation aus dem internen Netzwerk in das "Anmelde-Netzwerk" zu kontrollieren. Eine Lösung für dieses Problem ist der Einsatz einer zustandsbehafteten Filterregel. Diese wird in der Kette **Forward** erzeugt und leitet Pakete, welche zu einer Verbindung gehören, die aus dem "Anmelde-Netzwerk" initiiert wurde, auch in dieses zurück. Die Policy der Tabellen wird standardmäßig auf das Verwerfen nicht durch Regeln zugelassener Pakete gesetzt, damit können nur genehmigte Pakete den Filter passieren.
Die Freischaltung eines Supplicanten wird wie folgt erreicht. Nach einer erfolgreich durchgeführten Anmeldung wird eine Regel in der **Mangle-Prerouting** Kette erzeugt. Durch diese Regel wird die Source-MAC-Addresse als authentifizieriert und zugangsberechtigt klassifiziert. In den folgenden Instanzen wird unter Berücksichtigung dieser Klassifizierung das Paket entsprechend weitergeleitet. Ist diese Source-MAC-Addresse nicht durch diesen Mechanismus freigeschalten, werden die Pakete, von diesem Gerät verworfen. Diese Umsetzung erlaubt es, dass weitere Filterregeln für die Pakete

geprüft werden können. Dadurch ist es möglich, weitere Verfahren für die Authentifizierung zu nutzen. Wichtig ist nur, dass diese parallel existierenden Verfahren hierarchisch angeordnet und in den Filtertabellen abgebildet werden können. Im Kapitel 6 Abschnitt 6.4.1 werden die genutzten Regeln sowie die Syntax aufgezeigt.
Bei der Umsetzung dieses Sperrmechanismusses müssen Unterschiede in der LAN und WLAN Struktur betrachtet werden. Im WLAN ist der Einsatz dieser Lösung problemlos möglich. Im Gegensatz dazu wird im LAN ein schon erwähntes Umschalten des VLAN vorgenommen. Dies hat zur Folge, dass das Nutzergerät sich anschließend in einem anderen physischen Subnetz befindet. Dadurch erfolgt die Versendung der Pakete vom Nutzer nicht mehr über den "Anmelde-Server". Die Kontrolle der Verbindungen liegt somit beim "Firewall-Server". Für die Lösung dieses Problems stehen mehrerer Möglichkeiten zur Verfügung, welche folgend betracht und bewertet werden soll.
Die erste Möglichkeit ist eine Verteilung der Software des Authenticators. Das heißt, die Anmeldung erfolgt über den "Anmelde-Server" und folgend wird eine Übergabe an den "Firewall-Server" durchgeführt. Dabei werden Informationen über den Anmeldevorgang an eine zweite Instanz des Authenticators auf einem anderen Rechner übergeben. Dieser ist dann für die Kontrolle der Verbindungen sowie für die Prüfung der Verfügbarkeit des Nutzers verantwortlich. Ebenso muss der Authenticator ein entsprechendes Umschalten des VLAN's durchführen, was weitere Funktionalität erfordert. Dieses Verfahren erfordert ein hohes Maß an zusätzlicher Funktionalität des Authenticators, welche entsprechend implementiert werden muss. Ist diese Funktionalität vorhanden, kann der Authenticator problemlos in ein derartig getrenntes LAN eingesetzt werden.
Die zweite Möglichkeit ist das Erweitern der LAN-Struktur im "Anmelde-Netzwerk". Dabei ist es notwendig, dass der "Anmelde-Server" einen entsprechenden Zugang zum internen Netzwerk besitzt und diesen entsprechend für authentifizierte Nutzer freischalten kann. Probleme existieren dadurch, dass die eingesetzte Software des Portmanagers dies nicht berücksichtigt und dadurch Konflikte entstehen. Dieser Einsatz ist bei einer parallelen Nutzung der Systeme nicht zu empfehlen.
Eine letzte Möglichkeit ist eine Kombination der Funktionalität des Authenticators und des Portmanager. Der Portmanager besitzt ein zentrales für die Umschaltung nötiges Programm, welches über eine Netzwerkverbindung und ein einfaches Textprotokoll die nötigen Daten zur Abarbeitung empfängt. Das Programm erhält über diese Schnittstelle die nötigen Daten, um einen Nutzer in das "Firewall-Netzwerk" umzuschalten und anschließend die Verfügbarkeit zu kontrollieren. Das Ziel dieser Kombination der beiden System ist, dass der Authenticator sich dieser Funktionalität bedient und dadurch ein Einsatz in dieser LAN-Struktur erfolgen kann. Probleme bei einer solchen Lösung sind, dass der Authenticator nach erfolgter Anmeldung seine Aufgaben wie Reauthentifizierung und Freischaltung an die Software des Portmanagers abgibt und somit nicht die volle Funktionalität von 802.1x zur Verfügung steht. Weiterhin muss der Authenticator über die nötigen Zugangsdaten für einen Freischaltung verfügen. Da er jedoch diese Daten aus der Anmeldung des Nutzers nicht sammeln kann, müsste ein Account für den Authenticator angelegt werden, welcher als universeller Account zur Freischaltung dient. Dies erschwert jedoch die Analyse, welcher Nutzer sich zu welcher

Zeit angemeldet hat. So erfordert dies eine Kontrolle der Logdaten aller Systemkomponenten. Der Vorteil dieser Lösung ist, dass diese einfach umzusetzen ist und durch schon gewonnene Erfahrungen nahezu problemlos funktioniert. Ebenso ist die Möglichkeit eines Konflikts zwischen dieses beiden Systemen nahezu ausgeschlossen. Für den Einsatz eines Software-Authenticator im LAN wurde diese Möglichkeit auf Grund von geringen Konfliktsituationen realisiert. Die entsprechende Vorgehensweise wird bei der Erläuterung der nötigen Software im Kapitel 6 im Abschnitt 6.4.2 vorgestellt.

5.3.5 Nutzersoftware

Die Software, über welche der Nutzer verfügen muss, ist ein Supplicant und ein DHCP Klient für sein entsprechendes Betriebssystem. Der DHCP Klient ist in den heutigen System meist standardmäßig verfügbar. Für den Supplicant stehen mehrere Lösungen zur Verfügung. Die Tabelle 4.1 zeigt die Verfügbarkeit und die Eigenschaften dieser Supplicanten für die entsprechenden Betriebssysteme.
Bei der Auswahl eines Supplicanten sind einige Punkte zu berücksichtigen. So sollte dieser die gewählten Authentifizierungsverfahren für die umgesetzte 802.1x Umgebung besitzen. Das heißt, er sollte den Einsatz von EAP-TTLS mit dem inneren Verfahren PAP beherrschen, da diese die Grundlage in dieser Umgebung bilden. Zusätzlich sollte der Einsatz in unverschlüsselten Funknetzwerken möglich sein, da an der TU Chemnitz diese verwendet werden. Falls eine Nutzung der existierenden WEP-Funktionalität ermöglicht wird, entfällt diese Voraussetzung. Ein weiterer wichtiger Punkt ist die Nutzung einer Unicast-Adresse, um Probleme bei der Verwendung der Software in durch Switches getrennten Netzwerken zu vermeiden.
Unter Berücksichtigung dieser Voraussetzungen wird empfohlen, für Linux die Software *xsupplicant* des Open1x Projektes und für Microsoft Windows die Lösung *Wire1x* von Wireless Internet Research & Engineering (WIRE) Lab. einzusetzen, wobei bei *Wire1x* eine Erweiterung für den Einsatz einer Unicast-Adresse getroffen werden muss. Diese Erweiterung ist im Anhang C.2 vorgestellt. Eine Beispielkonfiguration sowie eine Liste der eventuell benötigten Softwarekomponenten für die Nutzung befindet sich im Anhang C auf Seite 83.
Ein Einsatz anderer Supplicanten kann ebenso erfolgen, jedoch sollte dieser im Vorfeld geprüft und ausgiebig getestet werden, damit Nutzer keine Probleme bei der Anwendung dieser Software haben.

Kapitel 6

Software-Authenticator

Wie bereits erwähnt, musste für den Einsatz der 802.1x Funktionalität eine spezieller Software-Authenticator entwickelt werden. Dieser sollte bestimmte Voraussetzungen für die Nutzung erfüllen. In den folgenden Abschnitten wird diese Softwarelösung detaillierter vorgestellt und der Aufbau sowie Eigenschaften der Implementierung und der Nutzung aufgezeigt. Ein wesentlicher Punkt ist dabei die Beschreibung der Funktionsweise zentraler Mechanismen.

6.1 Allgemeine Details

Der Software-Authenticator wurde für den Einsatz unter Linux entwickelt und in der Programmiersprache "**C**" geschrieben. Dazu wurden spezielle Fähigkeiten für die Nutzung in der 802.1x Umgebung der TU Chemnitz implementiert.

Alle entwickelten Module dieses Authenticators stehen unter der GNU General Public License (GPL) Version 2. Der Wortlaut dieser Lizenz kann auszugsweise in den Sourcecodedateien beziehungsweise unter *http://www.gnu.org/licenses/gpl.html* nachgelesen werden.

6.2 Aufbau und Funktionsweise

Die folgenden Abschnitte sollen eine kurze Übersicht über die bei der Entwicklung durchgeführte Modularisierung geben und die verwendeten Quelldateien aufzeigen. Alle nötigen Funktionen, welchen verwendet werden, tragen ein definierte Bezeichnung um die Lokalität der Deklaration und Implementierung einfach festzustellen. Den Aufbau dieser Namensgebung zeigt die Tabelle 6.1.

Durch diesen Aufbau ist das Finden von Funktionen, sowie eine Fehlersuche und die Integration von Erweiterungen einfach durchzuführen. Die vollständige Funktionsreferenz befindet sich im Anhang B auf Seite 74.

Funktionsname	**Gliederung**
_"8021x"_"Modul"_"Funktion"	8021x: Funktion aus diesem Projekt Module: Modulename (Dateiname: "8021x_"Module".h/c") Funktion: eigentlicher Funktionsname
_8021x_auth_init	8021x: Funktion aus diesem Projekt auth: Module: "auth" (Dateiname: "8021x_auth.h/c") Funktion: "init"

Tabelle 6.1: Aufbau der Funktionsnamen

6.2.1 Senden und Empfangen der Pakete

Für das Senden und Empfangen von Netzwerkpaketen werden zwei verschiedene Verfahren verwendet. Das erste wird für die Kommunikation zwischen dem Supplicanten und dem Authenticator benutzt, welche über die Verbindungsschicht stattfindet. Für das Senden der Pakete wurden Funktionen der "LIBNET" [51] verwendet. Diese Bibliothek erleichtert die Nutzung verschiedener Ebenen des Protokollstacks. Dadurch konnte die Übermittlung von Raw-Ethernet-Paketen einfach implementiert werden. Für den Empfang dieser Pakete wurde die Bibliothek "LIBPCAP" [52] verwendet. Diese besitzt Funktionen, um Pakete direkt vom Interface zu empfangen. Ebenso ist es möglich eine Vorauswahl der Pakete durch das Setzen einer Filterregel zu erreichen. Die standardmäßig gesetzte Regel zeigt die Tabelle 6.2. Diese Regel bewirkt das Empfangen von Paketen, welche den Ethernet-Protokolltyp "*0x888E*" besitzen. Dieser Type wird ausschließlich für die 802.1x Pakete genutzt.

Filterregel	ether proto 0x888e

Tabelle 6.2: Filterregel der *Libpcap*

Das zweite Verfahren, welches zum Einsatz kommt, dient der Kommunikation zwischen dem Authenticator und dem Authentication-Server, welcher ein RADIUS Server ist. Für den Austausch der Daten ist das RADIUS Protokoll [53] notwendig. Dieser erfolgt über eine UDP-Verbindung. Dafür werden die von Linux angebotenen Socketfunktionen genutzt.
Die Kommunikation beider Verfahren erfolgt verbindungslos. Bei eventuell auftretenden Paketverlusten werden durch das Timingverhalten der 802.1x Funktionalität eine neue Versendung dieser Pakete initiiert beziehungsweise entsprechende Abbruchmaßnahmen der Authentifizierung eingeleitet. Um die Abarbeitung nicht zu blockieren, werden die Funktionen, welche das Senden und den

Empfang realisieren im so genannten Non-Blocking-Mode betrieben. Die Prüfung, ob neue Pakete verfügbar sind, erfolgt zyklisch beim Durchlauf der Hauptabarbeitungsschleife.
Die folgende Auflistung zeigt die Module, welche Funktionen für die einzelnen Verbindungen zur Verfügung stellen.

- **8021x_nal.h/c**
  - Stellt Funktionen für die Kommunikation zwischen Supplicant und Authenticator bereit (Raw-Ethernet-Pakete).
- **8021x_radius.h/c**
  - Stellt Funktionen für die Kommunikation zwischen Authenticator und Authentication-Server bereit (UDP-Verbindung).

6.2.2 Paketerzeugung

Für die Paketerzeugung wurde entsprechend der verwendeten Protokolle mehrere Module implementiert. Die Funktionen erzeugen Pakete des gewünschten Typs mit entsprechenden Header- und Nutzdaten. Die folgende Liste zeigt diese Module und ihr Einsatzgebiet.

- **8021x_ethernet.h/c**
  - Funktionen zum Erzeugen von Ethernet-Paketen.
- **8021x_eapol.h/c**
  - Funktionen zum Erzeugen von EAPOL Paketen.
- **8021x_eap.h/c**
  - Funktionen zum Erzeugen von EAP Paketen.

Für die Verbindung zum Authentication-Server wird das RADIUS Protokoll benötigt. Die Funktionen zum Erzeugen der Pakete für diese Verbindung befinden sich in der Datei "**802.1x_radius.h/c**".
Die Komposition der Pakete, speziell der Pakete der Verbindungsschicht, erfolgt in der Datei "**8021x_tx.h/c**". Dieses Modul besitzt mehrere Funktionen, welche abhängig vom aktuellen Zustand des Supplicanten Pakete erzeugen und anschließend ein Versenden veranlassen.

6.2.3 Supplicant Informationen

Als Voraussetzung für diesem Authenticator stand die Eigenschaft, mehrere Supplicanten an einem Interface zu authentifizieren. Dies erforderte die Implementation einer dynamischen Datenstruktur, welche alle nötigen Parameter eines Supplicanten wie aktueller Zustand, Informationen über die

Identität sowie Timervariablen enthält. Die Implementation und Deklaration dieser Datenstruktur und der nötigen Funktionen sind in den Dateien "**8021x_supp.h**" und "**8021x_supp.c**" enthalten. Die Realisierung erfolgt auf Basis einer einfach verketteten Liste, welche zyklisch durchlaufen wird. Die Anzahl der Operationen, welche pro Zustand nötig sind, ist gering. Dadurch konnte diese einfache Datenstruktur ohne große Performanceverluste implementiert werden.

6.2.4 Authenticator Informationen

Die Informationen, welche der Authenticator zur Abarbeitung benötigt, gliedern sich in zwei Kategorien. Die erste ist die Festlegung konstanter Werte im Quellcode. Diese sind in der Datei "**8021x_global.h**" deklariert. Dort werden zum Beispiel Konstanten für Paketheaderdaten sowie für die 802.1x Funktionalität definiert. Die zweite Kategorie sind Daten, welche beim Start des Systems übergeben oder während der Laufzeit erzeugt werden. Für diese Daten steht eine entsprechende Datenstruktur zur Verfügung. Ebenso werden in dieser Struktur Informationen über die Schnittstellen, welche der Kommunikation dienen, gespeichert. Die Deklaration dieser Datenstruktur befindet sich in der Datei "**8021x_auth.h**".

6.3 802.1x Funktionalität

Der Software-Authenticator ist nach dem Standard für 802.1x entwickelt. Da jedoch dieser Standard nur für den Einsatz direkt an der Schnittstelle, entwickelt wurde, mussten einige Änderungen für die zentrale Anwendung realisiert werden. Diese betreffen speziell die Statemachines, welche folgend näher betracht werden. Eine Änderung an der Implementierung der Statemachines ist nicht notwendig. Das einzige, was einer Änderung der 802.1x Funktionalität bedarf, ist die Erweiterung für eventuelle neue Authentifizierungsverfahren. Der letzte Abschnitt zeigt die Möglichkeit dafür auf.

6.3.1 Statemachines

Für die Abarbeitung der Authentifizierung definiert der Standard 802.1x für den Authenticator mehrere Statemachines. Die Implementation dieser befindet sich in der Datei "**8021x_sm.c**". Die Funktionen besitzen keinen Zustandsspeicher, diese nötigen Informationen werden direkt in der Datenstruktur des Supplicanten gespeichert. Dadurch ist die Nutzung durch mehrere Supplicanten möglich. Ebenso kann diese Implementierung später für eine verteilte Lösung weiterentwickelt werden. Die folgenden Abschnitte stellen die implementierten Statemachines einzeln vor.

Authentication Statemachine

Die Authentication-Statemachine ist für die zentrale Steuerung eines an das System gebundenen Supplicanten nötig. Diese wird für die Aktivierung anderer Statemachines genutzt. Im Gegensatz zu der für die Hardware definierten Statemachine aus dem Standard wurden ein paar Änderungen für

die Softwarelösung eingebracht. Die folgende Auflistung gibt einen Überblick über die wichtigsten Eigenschaften eines Zustands an. Das Zusammenwirken ist in Abbildung 6.1 zu sehen.

- **INITIALIZE**
  - Variablen der Supplicant-Datenstruktur werden initialisiert
- **DISCONNECTED**
  - Dieser Zustand wurde gegenüber dem 802.1x verändert. Es existieren zwei verschiedene Abarbeitungvarianten. Dies war nötig, um die eigentliche Struktur der Statemachine beizubehalten.
  - Zustandsübergang von "INITIALIZE"
    - Der neue Supplicant wird mit weiteren Variablen initialisiert. Anschließend erfolgt ein Zustandsübergang in den "CONNECTED" Zustand.
  - Zustandsübergang von anderen Zuständen
    - Supplicant erhält ein EAP-Failure Paket.
    - Löschen der erfolgten Freischaltung und das Entfernen aus der dynamischen Datenstruktur des Supplicanten wird durchgeführt.
- **CONNECTED**
  - Wird ein neuer Supplicant in das System hinzugefügt, erfolgt ein schneller Übergang in diesen Zustand. Hier erfolgt das Senden das EAP-Request Identity Paketes. Folgt keine Antwort nach einer definierten Zeit, wird dies erneut versucht, bis die maximale Anzahl erreicht wurde.
- **AUTHENTICATING**
  - EAP-Response Identity Paket wurde erhalten.
  - Dieser Zustand aktiviert die Backend-Authentication Statemachine und wartet auf ein Ereignis, welches das Resultat der stattfindenden Authentifizierung zeigt.
- **AUTHENTICATED**
  - Authentifizierung erfolgreich.
  - Aktiviert Reauthentication Statemachine und Key-Transmit Statemachine, falls diese durch die Konfiguration nicht deaktiviert wurden.
  - Startet das externe Programm des Sperrmechanismusses.

- **ABORTING**
  - In diesen Zustand wird bei einem auftretenden Fehler (kein Authentifizierungsfehler) in der Backend-Authentication Statemachine gewechselt. Dies bewirkt anschließend einen Übergang in den "DISCONNECTED" Zustand, wo der Supplicant aus dem System entfernt wird.
- **HELD**
  - Bei einem aufgetretenen Authentifizierungsfehler folgt ein Wechsel in diesen Zustand. Die mögliche Authentifizierung wird für eine definierte Zeit blockiert, bevor ein neuer Versuch gestartet wird. Dieses Warten dient dem Rücksetzen verschiedener Parameter beim Authenticator und Supplicant.

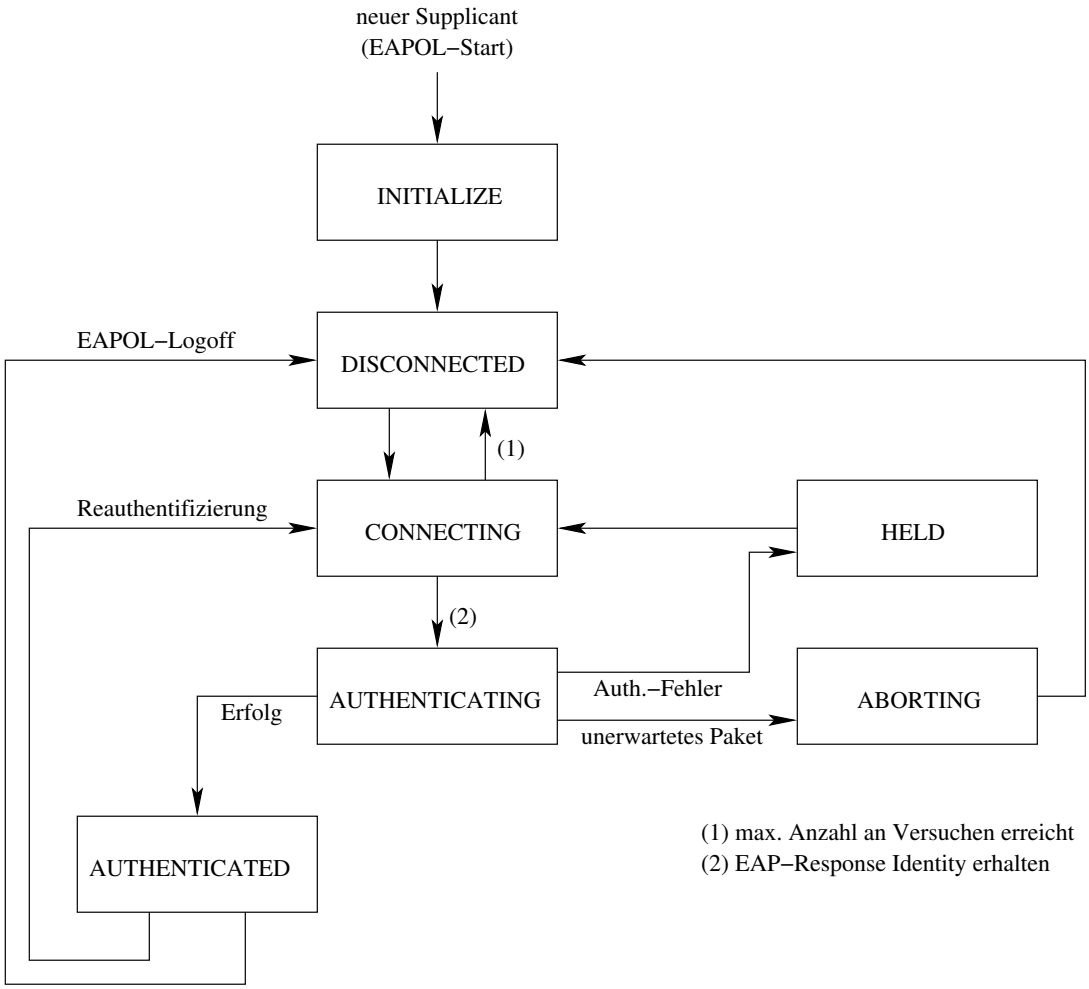

Abbildung 6.1: Authenticator Zustandsgraph

Der **INITIALIZE** und **DISCONNECTED** Zustand sind für den hardwaremäßigen Einsatz relevant. Diese werden nach dem Einschalten des Authenticators benötigt. So ist ein aktivierter Authenticator in einem Switch standardmäßig im **DISCONNECTED** Zustand und geht nach Erhalt

eines EAPOL-Start Pakets in den **CONNECTED** Zustand über. Bei der Softwarelösung ist die Abarbeitung dahingehend geändert, dass jeder Supplicant diese Statemachine besitzt und nicht der Authenticator. Der Supplicant wird erst nach Erhalt eines EAPOL-Start Pakets in das System aufgenommen. Dadurch musste eine Änderung für einen schnellen Übergang in den **CONNECTED** Zustand implementiert werden. Eine andere Umgehung hätte eine größere Änderung der Statemachine der, im Standard festgelegten, erfordert.

Backend-Authentication Statemachine

Diese Statemachine dient dem Ablauf der eigentlichen Authentifizierung des Supplicanten. Hier arbeitet der Authenticator lediglich als Proxy zwischen dem Supplicanten und dem Authentication-Server. Aktiviert wird diese Statemachine nach Erhalt eines EAP-Response Identity Pakets vom Supplicanten. Das folgende Listing beschreibt die einzelnen Zustände. Das Zusammenwirken ist in der Abbildung 6.2 zu sehen.

- **INITIALIZE**
  - Initialisierung einzelner Parameter für den Ablauf.
- **IDLE**
  - Zustand, wenn keine Authentifizierung stattfindet.
- **RESPONSE**
  - Hier werden Antworten vom Supplicant an den Authentication-Server gesendet und auf entsprechende Bestätigungen gewartet.
  - Zustandsübergang
    - Authentifizierung erfolgreich => in den "SUCCESS" Zustand (RADIUS-Success Nachricht)
    - Authentifizierung fehlgeschlagen => in den "FAILURE" Zustand (RADIUS-Failure Nachricht)
    - Zeitüberschreitung => in den "TIMEOUT" Zustand
- **REQUEST**
  - Pakete vom Authentication-Server werden an den Supplicanten zur Bearbeitung weitergeleitet.
  - Zustandsübergang
    - Antwort erhalten => zurück in den "RESPONSE" Zustand
    - Zeitüberschreitung => in den "TIMEOUT" Zustand

- **TIMEOUT**
  - Senden eines EAP-Failure Paketes an den Supplicanten.
  - Ereignis für den Übergang zum "ABORTING" Zustand in der Authentication Statemachine wird generiert.
- **FAILURE**
  - Senden eines EAP-Failure Paketes an den Supplicanten.
  - Ereignis für den Übergang zum "HELD" Zustand in der Authentication Statemachine wird generiert.
- **SUCCESS**
  - Senden eines EAP-Success Paketes an den Supplicant.
  - Ereignis für den Übergang in den "AUTHENTICATED" Zustand der Authentication Statemachine wird generiert.

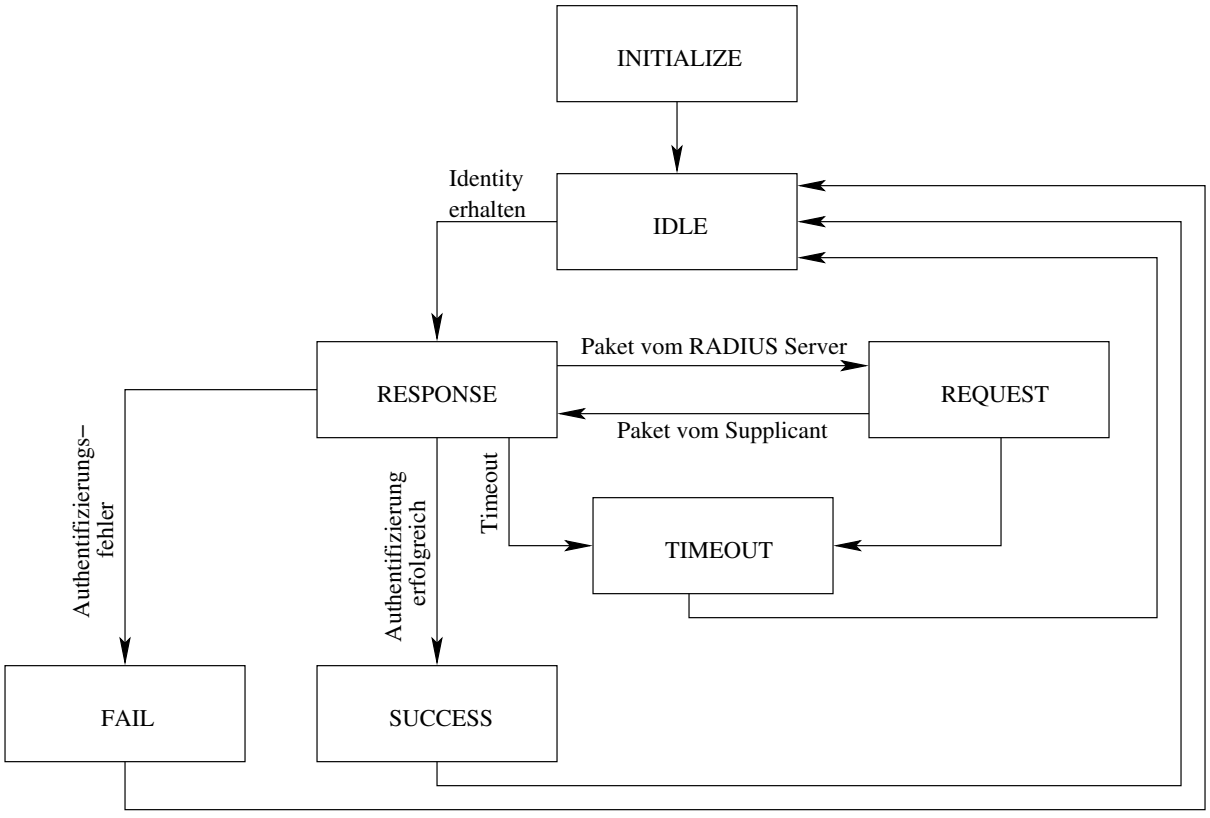

Abbildung 6.2: Backend-Authentication Statemachine

Reauthentication Statemachine

Die Reauthentication Statemachine besitzt lediglich zwei Zustände, einen Initialisierungs- und einen Reauthenticate-Zustand. Der Initialierungszustand setzt die Zeitvariable bis zur einer erneuten Authentifizierung. Nach Aktivierung dieser Statemachine wird der Initialisierungszustand verlassen und der Timer aktiviert. Nach Ablauf des Timers erfolgt ein Eintritt in den Reauthentication-Zustand. Dort wird ein Ereignis für die Authentication Statemachine erzeugt, um diese für die neue Authentifizierung zu veranlassen. Diese Reauthentifizerung bewirkt vorerst keine Änderung in den vorher getroffenen Einstellungen zur Freischaltung des Supplicanten.
Die Nutzung der Reauthentication Statemachine kann explizit in der Konfiguration ein- oder ausgeschaltet werden. Falls keine Nutzung erfolgt, können jedoch Probleme mit Supplicanten auftreten, welche sich nicht ordnungsgemäß vom System abmelden. Diese verbleiben in der dynamischen Datenstruktur, und ein Zurücksetzen der erfolgten Freischaltung wird nicht durchgeführt. Auf die Abschaltung sollte somit verzichtet werden, da dieser Mechanismus nicht nur das Problem eines fehlerhaften Verhaltens eines Supplicant behebt, sondern auch eine Kontrolle der Verfügbarkeit dessen vornimmt. Das Ausschalten ist lediglich für Testzwecke vorgesehen.

Key-Transmit Statemachine

Diese Statemachine wird zum Übermitteln der Schlüssel für Funknetzwerke mit WEP-Verschlüsselung genutzt. Die Nutzung kann in der Konfiguration explizit vor dem Start des Authenticators geschalten werden. Wird die Schlüsselübermittlung genutzt, so wird diese Statemachine nach erfolgreicher Authentifizierung und bei Erhalt der nötigen Informationen vom RADIUS Server aktiviert. Anschließend werden die Schlüssel an den Supplicanten übermittelt. Da sich der Authenticator nicht direkt im Access Point befindet, muss der nötige Schlüssel, welcher an den Supplicanten übermittelt wird, beim Start konfiguriert und im Access Point eingetragen sein. Für die Nutzung ist es weiterhin notwendig, dass der Access Point ebenso Verbindungen ohne gültigen Schlüssel zulässt.

6.3.2 Unterstütze Authentifizierungsverfahren

Der Authenticator unterstützt alle gängigen Authentifizierungsverfahren. Dazu gehören MD5, TLS, TTLS, PEAP, GTC und OTP. Für die Nutzung eines inneren Verfahrens gibt es keine Einschränkung. Es werden lediglich die äußeren Verfahren geprüft und entsprechend weitergeleitet. Eine Erweiterung der Auswahl an Verfahren kann in der Datei "**8021x_rx.c**" erfolgen. Diese beinhaltet Funktionen für die Analyse erhaltener Pakete. Dabei wird ebenfalls eine Kontrolle der Headerdaten durchgeführt. Um eine Erweiterung vorzunehmen, muss dem System die entsprechende Typkennung hinzugefügt werden. Das folgende Listing aus dem Quellcode zeigt einen Auszug aus der für eine Erweiterung notwendigen Funktion.

Datei: "**8021x_global.h**"

```
#define _8021X_EAP_TYPE_NAK_ 3                  /* only for response, RFC2284 (3.3) */
#define _8021X_EAP_TYPE_MD5_ 4                  /* MD5 challenge */
#define _8021X_EAP_TYPE_OTP_ 5                  /* One Time Password  */
#define _8021X_EAP_TYPE_GTC_ 6                  /* Generic Token Card */
#define _8021X_EAP_TYPE_TLS_ 13                 /* TLS Authentication */
#define _8021X_EAP_TYPE_TTLS_ 21                /* TTLS Authentication */
#define _8021X_EAP_TYPE_PEAP_ 25                /* PEAP Authentication */
```

Datei: "**8021x_rx.c**"

```
int _8021x_rx_eap_pktparse
(unsigned char *packet, struct _8021x_supp_supplicant **supplicants)
{
    ...

    if (eappkt->type == _8021X_EAP_TYPE_IDENTITY_) {
        ...
    } else if
        (
            /* allowd response pakets => transmit directly to radius server */
            (eappkt->type==_8021X_EAP_TYPE_NAK_)  ||
            (eappkt->type==_8021X_EAP_TYPE_MD5_)  ||
            (eappkt->type==_8021X_EAP_TYPE_OTP_)  ||
            (eappkt->type==_8021X_EAP_TYPE_GTC_)  ||
            (eappkt->type==_8021X_EAP_TYPE_TLS_)  ||
            (eappkt->type==_8021X_EAP_TYPE_TTLS_)  ||
            (eappkt->type==_8021X_EAP_TYPE_PEAP_)
        )
    {
            /* set response var ( state machine ) */
            supp->rxResp=1;
    }
    ...
}
```

6.4 Sperrmechanismus

Der Sperrmechanismus stellt einen zentralen Punkt des Software-Authenticators dar. Er verhindert die nicht authorisierte Nutzung von Ressourcen der TU Chemnitz sowie den Zugang zum Internet. Das eingesetzte Verfahren wurde bereits in Abschnitt 5.3.4 beschrieben. Die Umsetzung erfolgt durch die Nutzung externer Programmressourcen. Die Funktionalität ist somit unabhängig vom Software-Authenticator implementiert. Diese Auslagerung der Funktionalität bringt einige Vorteile mit sich. So ist durch eine externe Lösung eine Anpassung an verschiedene Netzwerkstrukturen und an eine Vielzahl verschiedener Technologien möglich.
Für die Nutzung dieser Funktionalität steht eine definierte Schnittstelle zwischen dem Authenticator und dem externen Programm zur Verfügung. Die Übergabe spezieller Werte an das externe Programm erfolgt über die Kommandozeile. Zur Nutzung dieses Verfahrens muss das externe Programm vier Funktionen zur Verfügung stellen. Diese sind wie folgt definiert.

- Initialisierungsfunktion
  - Übergabeparameter: "-i"
- Deinitialisierungsfunktion
  - Übergabeparameter: "-x"
- Funktion zum Hinzufügen eines Supplicanten
  - Übergabeparameter: "-a <MAC-Adresse des Supplicant>"
- Funktion zum Entfernen eines Supplicanten
  - Übergabeparameter: "-r <MAC-Adresse des Supplicant>"

Der Rückgabewert nach der Ausführung eines Programmteils sollte bei Erfolg "0" betragen und bei einem aufgetretenen Fehler einen beliebigen anderen Wert besitzen. Die Schnittstelle ist einfach gehalten, kann jedoch beliebig durch eine entsprechende Änderung in der Datei "**8021x_filter.c**" angepasst beziehungsweise erweitert werden.
Für den Einsatz wurden zwei externe Programme für den Sperrmechnismus erstellt. Dazu wurden die beiden vorgestellten Verfahren bei der Umsetzung im LAN und WLAN Bereich des Rechenzentrums in zwei PERL Skripten implementiert. Diese Implementierung der beiden Verfahren wird folgend näher vorgestellt.

6.4.1 WLAN

Bei WLAN kommt das Verfahren zum Einsatz, welches entsprechende Regeln in den "IP-Tables" des Rechners setzt. Die nötigen Konfigurationeinstellungen wurden direkt im Skript "**8021x_wlan.pl**" hinterlegt. Für das Filterprogramm des WLAN wurden folgende Parameter fest definiert, welche vor dem ersten Start an das eigentlichen Netzwerk angepasst werden müssen.

- Filterprogramm
  - $FILTER_PROG = "/sbin/iptables"
- Markierung der Pakete
  - $FILTER_MARK = 159
- Schnittstelle (Device) zum geschützten Netzwerk
  - $FILTER_OUT_DEV = "eth0"

Initialisierung

Bei der Initialisierung werden die statischen Regeln sowie die globale Eigenschaft für die Handhabung mit den Paketen gesetzt. Dafür werden zuerst alle nötigen Ketten, welche durch das System beeinflusst werden, auf die Policy "*DROP*" gesetzt, welche ein Wegwerfen der nicht berechtigten Pakete bewirkt. Anschließend wird eine Kette für authentifizierte Supplicanten in der **MANGLE** Tabelle erzeugt, sowie eine Referenz aus der **PREROUTING** Kette auf diese gesetzt.

$FILTER_PROG -t mangle -N 8021X_AUTH
$FILTER_PROG -t mangle -I PREROUTING 1 -j 8021X_AUTH

Zusätzlich zu dieser Regel wurden folgende Regeln in die **FORWARD** Kette eingetragen, wobei die Reihenfolge zu beachten ist. So wurde die Regel für das Weiterleiten von Paketen authentifizierter Nutzer an die erste Stelle gesetzt, da dass System für die 802.1x Authentifizierung an unterster Stelle einsetzt. Die zweite Regel kann an einer beliebigen Stelle der aktuellen Filterregeln gesetzt werden. Diese ist für die Kommunikation in die rückwärtige Richtung notwendig.

$FILTER_PROG -I FORWARD 1 -m mark –mark $FILTER_MARK -j ACCEPT
$FILTER_PROG -I FORWARD 2 -m state –state ESTABLISHED,RELATED -j ACCEPT

Abschließend muss der Zugang zum internen Netzwerk noch freigeben werden. Dazu wird mit Hilfe von NAT eine entsprechende Transformation der Sender-IP-Adresse vorgenommen.

Der hier eingesetzte Mechanismus bewirkt ein Umschreiben der Sender-IP-Adressen von allen Paketen der verschiedenen Supplicanten in die des zentralen Rechners. Eine Implementierung, welche eine 1:1 Übersetzung durchführt, wäre ebenso denkbar.

```
$FILTER_PROG -t nat -I POSTROUTING 1 -o $FILTER_OUT_DEV
                                -m mark –mark $FILTER_MARK -j MASQUERADE
```

Deinitialisierung

Die Deinitialisierung ist einfach gehalten. Es werden alle vorher gesetzten Regeln gelöscht sowie alle Einträge aus der Kette für die authentifizierten Supplicanten entfernt und anschließend gelöscht. Eine Rücksetzen der Policy erfolgt ebenso. Hier können jedoch Probleme mit anderen eingesetzten Authentifizierungsverfahren auftreten. Es ist möglich, dass können diese durch die veränderte Policy in ihrer Funktion beeinträchtigt werden.

Hinzufügen eines Supplicanten

Das externe Filterprogramm erhält bei dieser Funktion zusätzlich die MAC-Adresse des Supplicanten. Die Regel, welche anschließend in den IP-Tables gesetzt wird, bedient sich der Funktionalität, Pakete der Verbindungsschicht zu prüfen. Dadurch können alle vom Supplicanten gesendeten Pakete klassifiziert und später entsprechend weitergeleitet werden.

```
$FILTER_PROG -t mangle -A 8021X_AUTH -m mac –mac-source <MAC-Adresse> -j MARK –set-mark $FILTER_MARK
```

Entfernen eines Supplicanten

Beim Entfernen eines Supplicanten wird ebenfalls seine MAC-Adresse mit übergeben. Diese ist eindeutig und kann als Merkmal für die Regel dienen, welche gelöscht werden soll. Die folgende Syntax zeigt den entsprechenden Aufruf.

```
$FILTER_PROG -t mangle -D 8021X_AUTH -m mac –mac-source <MAC-Adresse> -j MARK –set-mark $FILTER_MARK
```

6.4.2 LAN

Im LAN der TU Chemnitz kann das Verfahren vom WLAN nicht genutzt werden. Die Trennung der VLAN's verhindert dies. Jedoch existieren bereits vorgestellte Ausweichverfahren. Dazu gibt der Authenticator nach erfolgter Authentifizierung die weiterführende Kontrolle an den durch den Portmanager bereitgestellten Mechanismus ab. Dazu müssen einige Parameter über eine TCP-Verbindung dem entsprechenden Programm übergeben werden. Eine kurze Beschreibung der vier

zur Verfügung stehenden Funktionen wird im folgenden vorgestellt. Ebenso wie im Skript für den WLAN-Bereich werden die Konfigurationsdaten für dieses Verfahren direkt im Programm definiert. Die benötigten Daten für die Umschaltung auf das Portmanager-System, welche im Skript "**8021x_pm.pl**" implementiert wurden, sind folgende.

- Portmanager IP-Adresse
  - $PM_IP = "127.0.0.1"
- Portmanager TCP Port
  - $PM_PORT = "1011"
- Nutzername (spzieller Account)
  - $PM_USER = "username"
- Passwort (spezieller Account)
  - $PM_PASS = "password"
- Firewall-Server (Netzwerk)
  - $PM_NETC = "default"

Initialisierung/Deinitialisierung

Die Initialisierung und Deinitialisierung wird nicht verwendet. Hier kann später noch zusätzliche Funktionalität implementiert werden.

Entfernen eines Supplicanten

Das Entfernen eines Supplicanten wird nicht durch das Filterprogramm realisiert. Wie bereits erwähnt erfolgt eine Übergabe an das Portmanager-System. Dieses prüft anschließend die Existenz des Nutzers über seine eigene Funktionalität und entfernt ihn nach seiner Abmeldung.

Hinzufügen eines Supplicanten

Im Gegensatz zu WLAN erfordert das Hinzufügen eines Supplicanten im LAN mehr Funktionalität. So ist für die Übergabe an den Portmanager die IP-Adresse des Supplicanten nötig. Diese ist dem System nicht bekannt, da die Kommunikation über die Verbindungsschicht stattfindet. Eine Ermittlung ist somit notwendig. Anschließend müssen alle benötigten Parameter an den Portmanager übergeben und auf eine entsprechende Antwort gewartet werden.
Für die Ermittlung der IP-Adresse können mehrere Verfahren angewendet werden. Bei diesem Skript wird durch das Auslesen des ARP-Caches die IP-Adresse über die MAC-Adresse ermittelt. Dies kann jedoch nur erfolgen, wenn der Nutzer schon während der Authentifizierung eine

IP-Adresse durch den DHCP erhalten hat. Dies ist beim Einsatz des zentralen Authenticators in Rechnernetzwerk der TU Chemnitz standardmäßig gegeben, da die parallel existierenden Verfahren ebenso auf dieses Adresse angewiesen sind. Anschließend an diese Ermittlung erfolgt eine Kommunikation mit dem Portmanager. Dazu werden diesem die entsprechenden Werte aus der folgenden Liste übergeben.

- HELO <IP-Adresse des Supplicanten>
- USER <Nutzername zur Umschaltung>
- PASS <Passwort>
- NETC <Angabe in welches Netzwerk umgeschalten wird>
- AUTH
  - Startet den Vorgang zur Umschaltung.

Wichtig bei der Nutzung dieses Filtermechanismusses ist die Bereitstellung eines Accounts für die Übermittlung an den Portmanager, welcher die entsprechende Berechtigung für eine Freischaltung besitzt. Die vom Nutzer gesendeten Daten können nicht verwendet werden, da der Authenticator nicht in die Kenntnis dieser gelangen kann.

6.5 Einsatz

Für den Einsatz dieser Software sind einige Voraussetzungen notwendig, welche folgend vorgestellt werden. Ebenso werden die Installation, die Konfiguration der Start und das Beenden des System erläutert.

6.5.1 Voraussetzungen

Für die Nutzung der Software müssen einige Voraussetzungen erfüllt sein. So sind verschiedene zusätzlich installierte Entwicklungspakete notwendig. Weiterhin ist der Einsatz nur unter Linux möglich. Die benötigten Voraussetzungen werden in der folgenden Liste aufgezeigt.

- Linux (Kernel >= 2.4.x, GLIBC >=2.3.2)
- LIBNET >=1.0.2
- LIBPCAP >=0.7.2
- OpenSSL >=0.9.7a
- IPTABLES >=1.2.8

Zusätzlich sollte für den Betrieb ein Rechner mit zwei Netzwerkschnittstellen zur Verfügung stehen. Es ist zwar möglich, die Aliaskonfiguration einer Netzwerkkarte unter Linux zu nutzen, das heißt, die Verwendung logischer Schittstellen, jedoch sollte zur Sicherheit eine physische Trennung erfolgen.

6.5.2 Kompilierung, Installation und Start

Für die Kompilierung steht ein "Makefile" zur Verfügung. Dieses erzeugt auf Grund der Abhängigkeiten das ausführbare Programm. Eine Installation erfolgt jedoch nicht. Dies sollte eigenverantwortlich vom Administrator durchgeführt werden. Der Start des Systems kann durch zusätzliche Mechanismen wie eines Init-Skriptes beim Start des Betriebssystems erfolgen. Wichtig ist, dass alle nötigen Abhängigkeiten sowie die Verfügbarkeit des Netzwerkes bereits aktiviert wurden. Der Start erfolgt über einen einfachen Programmaufruf mit der Angabe nötiger Parameter, welche noch im Abschnitt 6.5.3 vorgestellt werden. Ebenso benötigt der auf Grund des Zugriffs auf die Schnittstelle zur Versendung von Ethernet Paketen besondere Rechte. So sind unbedingt "**ROOT**" Rechte erforderlich. Die Ausführung der Software kann in einer geschützten Umgebung erfolgen, da der Zugriff nur auf die Konfigurationsdatei notwendig ist.

6.5.3 Konfiguration

Die Konfiguration kann auf zwei verschiedenen Wegen durchgeführt werden. Zu einem ist es möglich, alle wichtigen Parameter für die Ausführung auf der Kommandozeile zu übergeben. Dabei werden für weitere Einstellungen Standardwerte verwendet. Eine weit aus tiefer gehende Konfiguration kann mit Hilfe eine Konfigurationsdatei erreicht werden. Eine mögliche Konfiguration mit allen Werte befindet sich im Anhang B.2. Die Tabelle 6.3 gibt Aufschluss über die möglichen Werte, deren Standardbelegung und beschreibt die Wirkung dieser.

Kommandozeile ./8021x_auth
-a <Radius IP-Address>
-s <Radius secret>
-p <Radius port>
-i <Supplicant device>
-d <Debuglevel 0-5>
-c <Configfile>

Variable	Werte	Voreinstellung	Beschreibung
DebugLevel	0-5	1	Schalte Debugging und Logging im entsprechenden Level ein.
LOG	Dateiname	stdout	Definiert das Ziel der Ausgabe von Logmeldungen.
DEBUG	Dateiname	stdout	Definiert das Ziel der Ausgabe von Debugmeldungen.
WARNING	Dateiname	stdout	Definiert das Ziel der Ausgabe von Warningmeldungen.
ERROR	Dateiname	stderr	Definiert das Ziel der Ausgabe von Errormeldungen.
PACKET	Dateiname	stdout	Definiert das Ziel der Ausgabe für den Packetdump.
Device	Linux Devicename	keine	Interface, an welchem sich die Supplicanten authentifizieren.
PromiscousMode	0/1	1	Empfängt Pakete im PromiscousMode.
CaptureFilter	String	ether proto 0x888e	Filterstring für die Libpcap.
RadiusIP	IP-Adresse	keine	IP-Addresse des RADIUS Servers.
RadiusPort	UDP-Port	keine	UDP-Port des RADIUS Servers.
RadiusSecret	String	keine	Geheimnis zur Kommunikation mit dem RADIUS Server.
CalledStationId	String	8021x Software Authenticator	Identifier String des Authenticators.
CallingStationId	String	8021x Supplicant	Identifier String des Supplicanten.
ReAuthenticate	0/1	1	Einschalten der Reauthentifikation
PortControl	AUTO	AUTO FORCE_AUTHORIZED FORCE_UNAUTHORIZED	AUTO=Authentifizierung erforderlich FORCE_AUTHORIZED=Zugang ohne Authentifizierung möglich FORCE_UNAUTHORIZED=Kein Zugang möglich. (Port gesperrt)
SupplicantRequestTime	0-65535	10	Zeit, welche der Supplicant besitzt, um zu antworten (in Sekunden)
RadiusRequestTime	0-65535	10	Zeit, welche der RADIUS Server besitzt, um zu antworten (in Sekunden)
ReAuthenticateTime	0-65535	120	Zeit bis zur Reauthentifikation (in Sekunden)
KeyTxEnabled	0/1	0	Schaltet des Versenden der EAPOL-Key Pakete für WEP gesicherte Funknetzwerke ein.
Key	String von HEX-Werten	alles ”0”	Definierter Schlüssel der Access Points, welcher übermittelt wird.
KeyLength	5/13	5	Schlüssellänge des WEP-Schlüssels (5=40bit, 13=108bit)
FilterProgram	ausführbare Datei	./filter/8021x_fw.pl	Definiert das externe Filterprogramm

Tabelle 6.3: Konfigurationsmöglichkeiten für den Software-Authenticator

6.5.4 Betrieb/Beendigung

Während des Betriebes können keine Änderungen an der Konfiguration vorgenommen werden. Eine Möglichkeit ist die Implementation einer Schnittstelle zur Konfiguration zur Laufzeit, wobei dies nicht unbedingt notwendig ist. Bei einem Abbruch, welcher durch Fehler hervorgerufen wird, werden gegebenfalls gesetzte Regeln nicht aus den "IP-Tables" gelöscht. Beim erneuten Start erfolgt zwar eine Prüfung diesbezüglich, jedoch sollte ein Abbruch des Systems zusätzlich kontrolliert werden. Für die Beendigung des Programms stehen mehrere Möglichkeiten zur Verfügung. Bei einem Start von der Kommandozeile aus, bei dem das Programm im Vordergrund ausgeführt wird, kann ein Schließen durch "**CRTL-C**" erreicht werden. Dies bricht die Bearbeitung nicht ab, sondern beendet die Ausführungsschleife und entfernt alle stattgefundenen Einträge. Wird diese Software im Hintergrund gestartet, ist dies nicht möglich. Dafür kann durch Senden verschiedener Signale, auf welche der Authenticator reagiert, eine Beendigung erfolgen. Die Signale zum Beenden des Systems sind "**SIGINT**", "**SIGQUIT**" und "**SIGTERM**". Diese können zum Beispiel über den Befehl "*kill*" an die Software gesendet werden. Ein anderer Abbruch als die vorgestellte Möglichkeiten erreicht keine gezielte Beendigung.

6.6 Debugging

Für die Debugausgabe stehen mehrere Ebenen zur Verfügung, welche über die Einstellung des **DebugLevel** Parameters aktiviert werden. Diese verschieden Ebenen bauen aufeinander auf. So ist durch das Schalten der dritten Stufe die zweite und erste weiterhin aktiviert. Die Ausgabe der Debugdaten erfolgt nach den Angaben in der Konfigurationsdatei oder entsprechend der Standardausgabewerte. Die Liste der möglichen Einstellungen sowie die angezeigten Informationen und die Konfigurationvariablen für die Ausgabe werden in folgender Tabelle aufgezeigt.

DebugLevel	**Information über**	**Ausgabe**
0	keine Ausgaben/kein Logging	keine
1	Loginformationen über die An- und Abmeldung eines Supplicanten	LOG
2	Allemeine Debuginformationen über Authentictor und Supplicanten	DEBUG/ WARNING/ ERROR
3	Debuginformationen über die Statemachines	DEBUG/ WARNING/ ERROR
4	Alle Debugausgaben (zusätzliche Information über Datenstrukturen und Pakete)	DEBUG/ WARNING/ ERROR
5	Paket Dump (Hexadezimale Speicherung empfangener und gesendeter Pakete)	PACKET

Kapitel 7

Diagnosemöglichkeiten

Für die Diagnose einer 802.1x Umgebung stehen nur wenige Möglichkeiten zur Verfügung, die sich meist nur auf bestimmte Teile dieser Struktur anwenden lassen. Das Problem für eine vollständige Diagnose liegt im definierten Aufbau einer solchen Umgebung. So wird der Test eines Authenticators erschwert, da dieser sich meistens direkt im Netzwerkgerät wie Switch oder Access Point befindet, und dadurch eine Diagnose nur durch die direkte Verbindung möglich ist. Ebenso existieren Probleme bei eventuell eingesetzten Verfahren zur Verschlüsselung einer Funkverbindung. Zusätzlich werden zwei Kommunikationsverbindungen auf unterschiedlichen Ebenen des Protokollstacks genutzt, was zur Folge hat, dass diese Verbindungen beim Einsatz von Hardwareauthenticatoren schwer über eine Schnittstelle getestet werden können. Ein Ausnahme stellt der Software-Authenticator dar, welcher im Zusammenhang mit dieser Arbeit entwickelt wurde. Dieser kann nahezu beliebig im Netzwerk eingesetzt werden und bietet somit mehr Möglichkeiten für Diagnosen.
In den folgenden Abschnitten sollen kurz verschiedene Verfahren, welche auch im Zusammenhang mit dieser Arbeit genutzt wurden, vorgestellt werden. Der Einsatz ist, wie einleitend beschrieben, nur unter bestimmten Voraussetzungen möglich. Diese werden in den einzelnen Abschnitten jeweils vorgestellt.

7.1 Protokollanalyse

Für die Diagnose einer stattfindenden Authentifizierung ist es notwendig, die Pakete, welche während des Prozesses ausgetauscht werden, zu analysieren. Diese Pakete sind nicht überall mitlesbar. Es müssen entsprechende Softwarelösungen zum Mitlesen der Pakete auf der Plattform des Supplicanten oder beim Authentication-Server genutzt werden. Eine Möglichkeit zur Analyse beider Kommunikationsverbindungen ist der Einsatz eines Repeaters. Damit ist es möglich, alle Teilnehmer über dieses Netzwerkgerät kommunizieren zu lassen und durch den Anschluss eines zusätzlichen Rechners und unter Verwendung einer entsprechenden Software für diese Diagnose alle Pakete zu empfangen, um den zeitliche Ablauf der Authentifizierung nachzuvollziehen. Die gleiche Funktionalität stellt der Software-Authenticator zur Verfügung. Er besitzt die Möglichkeit,

alle gesendeten und empfangen Pakete zu speichern. Ein Problem bei diesem Einsatz ist, dass die Daten lediglich in hexadezimaler Form verfügbar sind. Somit muss eine Auswertung der Pakete durch den Nutzer selbst erfolgen. Diese Möglichkeit kann für kleine Analysen eingesetzt werden. Die nötigen Informationen für die Auswertung der vom Software-Authenticator gespeicherten Daten befindet sich im Anhang A auf Seite 67 dieses Dokumentes. Zusätzlich ist wichtig, dass der Ablauf einer solchen Authentifizierung, welcher im Kapitel 3 Abschnitt 3.3 vorgestellt wurde, bekannt ist, da die meistens Problem bei der Authentifizierung durch verwendete Authentifizierungsverfahren auftreten.
Die Auswertung der Nutzerdaten kann vielfach nicht erfolgen, da bei den Verfahren Hashwerte beziehungsweise sichere Kanäle genutzt werden. Eine Möglichkeit, um eine TLS-Verbindung zu entschlüsseln, wäre der Einsatz eines TLS-Proxies für das Authentifizierungsverfahren. Dieser müsste sich direkt im Authenticator befinden. Eine Erweiterung des Software-Authenticators wäre diesbezüglich möglich. Die Funktionalität sollte jedoch nur für Testzwecke verwendet werden, da der Authenticator nach dem Standard nicht an die Authentifizierungsdaten des Nutzers gelangen kann und auch nicht sollte.
Als Softwarelösungen für das Mitlesen der Pakete stehen eine Vielzahl an Programmen zur Verfügung. Die wesentlichen und häufig genutzten Vertreter sind *TCPdump* und *Ethereal*. Dabei biete *Ethereal* eine detailliert Analyse der Daten an. So werden empfangene Pakete bereits für eine bessere Verständlichkeit aufbereitet.

7.2 Authentifizierungsverfahren

Für die Diagnose der Authentifizierungsverfahren stehen kaum Möglichkeiten zur Verfügung. Um ein solches Verfahren zu testen, ist es wichtig, dass der Authentication-Server dies für alle Formen unterstützt, welche das Verfahren nutzt. Diese unterschiedlichen Formen kommen durch den Einsatz interner Protokolle zu stande wie zum Beispiel bei TTLS. Ein Analyse der Daten ist jedoch auf Grund durchgeführter Sicherungsverfahren schwierig. Eine mögliche Auswertung, kann somit lediglich am Authentication-Server erfolgen, da der Authenticator keine Möglichkeit besitzt, die Daten zu analysieren. Eine Auswertung beim Supplicant ist ebenso nicht möglich.
Die Diagnose beziehungsweise die Analyse hängt somit stark vom verwendeten Authentication-Server ab. Eine starke Funktionalität für dieses Vorgehen stellt der FreeRADIUS Server bereit. Dieser besitzt die Möglichkeit, eine detaillierte Ausgabe von Logdaten bezüglich des Authentifizierungsverfahrens durchzuführen. Ebenso wie das Verfolgen der einzelnen Schritte einer Authentifizierung kann dieser Server auch die Nutzerdaten, welche durch eine eventuell gesicherte Verbindung übertragen wurden, ausgeben.

7.3 Authenticator

Speziell bei eingesetzten Authenticatoren in Netzwerkgeräten stehen Schnittstellen zur Konfiguration und zur Diagnose zur Verfügung. Diese besitzen meistens eine eigene Struktur und sind vom Hersteller abhängig. Eine einheitliche Möglichkeit stellt der Standard 802.1x zur Verfügung. In diesem werden entsprechende Merkmale für die Diagnose und Konfiguration eines Authenticators über ein SNMP Interface definiert. Dieses kann, wenn es in der Hardware verfügbar ist, einen Aufschluss über den Ablauf der Authentifizierung an einem Port geben. Die nötige Management Informationbasis (MIB) befindet sich als Textdatei auf der beigefügten CD.
Die Softwarelösungen für Authenticatoren lassen sich im Gegensatz zu den Hardwarelösungen nicht über solche Schnittstellen bedienen oder analysieren. Bei diesen steht lediglich die vom Programmierer implementierte Debugausgabe zur Verfügung.
Ein Test dieser Authenticatoren setzt die Verfügbarkeit eines RADIUS Servers sowie eines funktionsfähigen Supplicanten voraus. Eine spezielle Vorgehensweise bei einer verteilten 802.1x Struktur, welche auf der Grundlage des entwickelten Software-Authenticators basiert, kann durch das gezielte Ansprechen eines dieser Authenticatoren durch die Verwendung einer Unicast-Adresse erfolgen. Die beste Möglichkeit ist jedoch, dass direkte Verbinden mit dem Authenticator zum Test, da dies Probleme durch die Netzwerkstruktur und eine eventuelle Filterung von Paketen ausschließt.

Kapitel 8

Fazit

Abschließend zu dieser Arbeit sollen einige wichtige Punkte, Probleme und Eigenschaften kurz zusammengefasst werden. Es erfolgt eine Beurteilung der durchgeführten Tests beim Einsatz einer 802.1x Umgebung an der Technischen Universität Chemnitz. Zusätzlich zu diesem Einsatz soll ein Ausblick auf mögliche Verbesserungen des Software-Authenticators, welche sich erst nach der Entwicklung feststellen ließen, gegeben werden

8.1 Zusammenfassung der Tests

Im Rahmen dieser Arbeit wurden verschiedene Tests mit unterschiedlichen Hard- und Softwarelösungen durchgeführt. Die verwendete Hardware verfügte zum Teil über einen eigenständigen Authenticator für die Authentifizierung. Alle Produkte, welche im Kapitel 4 vorgestellt wurden, konnten mit Erfolg getestet, jedoch nicht in jeder Umgebung genutzt werden.
Bei der Nutzung des Software-Authenticators aus dieser Arbeit in geteilten Netzwerken wurden Probleme, welche bereits erwähnt wurden, gefunden. Die Weiterleitung der EAPOL Pakete durch die Switches war ein hauptsächlicher Grund dieser Probleme im LAN. Im WLAN trat diese Filterung der Pakete ebenso auf, jedoch war die Filteregel, welche genutzt wurde, eine andere. So erfolgte durch den Access Point mit 802.1x Unterstützung das Filtern auf Basis des Ethernet-Protokolltyps und nicht wie in Switches anhand der Mulitcast-Adresse. Dies hatte zur Folge, dass eine einfache Lösung wie die Verwendung einer Unicast-Adresse das Problem nicht beheben konnte. Eine für die Tests realisierte Lösung war das Einspielen einer älteren Firmeware des Access Points ohne die 802.1x Funktionalität. Diese Probleme machten die Tests nur unter bestimmten Kombinationen möglich.
Von den Authentifizierungsverfahren wurden, die bereits im Kapitel 3 vorgestellte Verfahren getestet. Da diese jedoch unterschiedliche Protokolle zur Übertragung der eigentlichen Nutzerdaten verwenden, konnte keine einheitliche Lösung der Konfiguration des Authentication-Servers gefunden werden. Die größte Flexibilität besitzt das Verfahren, welches für den Einsatz an der TU Chemnitz vorgesehen wurde. Die Nutzung kann mit jeder vorgestellten Supplicantensoftware erfolgen.

Abschließend soll noch erwähnt werden, dass alle getesteten System ihre Funktionalität ohne Probleme zur Verfügung stellten und ein Einsatz dadurch entsprechend der genutzten Netzwerksumgebung möglich war.

8.2 Standard 802.1x

Der Standard 802.1x bietet ein sicheres System zur Authentifizierung und der damit stattfindenden Zugangskontrolle. Die nötige Software ist einfach anzuwenden und bietet durch gesicherte Verfahren bei der Authentifizierung einen hohen Schutz der durch den Nutzer gesendeten Daten. Ein Problem ist die Verfügbarkeit zusätzlicher Servertechnologie, was wiederum einen erhöhten Administrationsaufwand hervorruft. Sind diese zusätzlichen Server bereits vorhanden, oder sie verursachen keinen allzu hohen Aufwand bei der Bereitstellung und die eingesetzte Hardware unterstützt dieses Verfahren, sollte der Einsatz erfolgen. Der Einsatz von Hardware wie zum Beispiel Switches mit Authenticator oder Access Points mit der Verwendung von WPA oder 802.11i, welche diesen Standard nutzen, bilden in Verbindung mit einem RADIUS Server eine fast vollständig sichere Lösung für ein Zugangsmanagement. Ebenso ist der Einsatz der Schlüsselgenerierung in Verbindung mit den EAPOL-Key Paketen ein wesentlicher Vorteil dieses Standards. Durch die Verwendung dieser Technologie lassen sich relativ einfach sichere und einheitliche Verfahren zur Zugangssteuerung realisieren.
Ein Problem ist jedoch bei der Anwendung dieser Technologie das Fehlen einer Möglichkeit zur Realisierung von Nutzerklassen, welche in verschiedenen Institutionen Einsatz finden. Die Authenticatoren in den Hardwaresystemen unterstützen meist nur eine Freischaltung oder Sperrung des Zugriffsports. Ein bedingter Zugang erfordert zusätzliche Maßnahmen durch andere Schutzmechanismen.

8.3 Einsatz an der TU Chemnitz

Der geplante Einsatz im Rechnernetzwerk der TU Chemnitz kann nur bedingt erreicht werden. Die Probleme, welche dafür verantwortlich sind, werden nicht nur durch die eingesetzte Hardware hervorgerufen. Diese könnten durch eine einfache Umstrukturierung behoben werden. Eigentliche Probleme bereiten die bereits eingesetzten Verfahren zur Authentifizierung. So ist die parallele Existenz der verschiedenen Verfahren nicht vollständig mit der 802.1x Technologie einsetzbar. Die verwendeten Strukturen wie die Nutzung der verschiedenen VLAN's im LAN oder Implementierungen in der WLAN-Software können Probleme mit dem Software-Authenticator hervorrufen. Ebenso ist eine spätere Nutzung nur durch bestimmte Softwarelösungen für die Supplicanten möglich. Diese müssen besondere Eigenschaften besitzen. Der Einsatz kann somit nur unter bestimmten Bedingungen erfolgen und nicht universell für alle Systeme genutzt werden.
Ein Möglichkeit, die notwendigen Anforderungen an einen Supplicant zu erreichen, wäre die Implementierung einer solchen Software. Die Anpassung einer existierenden Lösung, wie dies bei dieser

Arbeit erfolgt ist, wäre ebenso denkbar.
Die Umsetzung einer solchen Umgebung fordert somit einen zusätzlichen Aufwand, welcher schwer bezüglich der Wirtschaftlichkeit abgeschätzt werden kann. Somit ist der Einsatz dieses Verfahrens mit einem zentral geschaltenen Authenticator nicht zu empfehlen.
Eine Verbesserung könnte jedoch durch die Verwendung der Authenticatoren in der Hardware erfolgen. Dadurch kann nicht nur eine einheitliche Struktur zur Authentifizierung geschaffen werden, sondern es bestehen bei diesem Einsatz nur wenige Restriktionen, was die Supplicantensoftware des Nutzers betrifft. Ein wesentliches Problem, welches in diesem Zusammenhang berücksichtigt werden muss, ist, dass alle bereits existierenden Verfahren deaktiviert werden müssen, da diese durch das vorgeschaltete 802.1x Verfahren nicht mehr nutzbar sind. Dazu müssen zusätzliche Maßnahmen für die Möglichkeit von bedingten Zugangslösungen, das heißt, durch verschiedene Nutzerklassen und die differenzierte Bereitstellung für die Nutzung öffentlicher und privater IP-Adressen realisiert werden, da dies die Hauptmerkmale der momentan verwendeten Lösungen sind. Ein Verzicht auf diese Eigenschaften wäre für die Umsetzung und Nutzung dieser Technologie empfehlenswert.

8.4 Software-Authenticator

Der Software-Authenticator ist für den Einsatz in Netzwerken, welche diesen ermöglichen, geeignet. Jedoch existieren Punkte, welche eine Verbesserung benötigen könnten. So ist bei starker Nutzung dieser Lösung eine Weiterentwicklung zu einem Mehrprozesssystem sinnvoll, da durch dies die Lastverteilung und Kapselung einzelner Supplicanten erreicht werden kann. Die momentan existierenden Module sind für eine solche Erweiterung bereits vorgesehen. Damit verbunden ist die Verbesserung des Timingverhaltens bei der Abarbeitung der Statemachines.
Als weitere Maßnahmen wären die Implementation einer Schnittstelle zur dynamischen Konfiguration, oder zur Abfrage vom Statusinformationen während der Laufzeit des Systems denkbar. Eine Möglichkeit dazu wäre die Realisierung einer SNMP Schnittstelle, welche diese Funktionalität unterstützt.

Anhang A

Paketanalysedaten

A.1 EAP

Die folgenden Abschnitte beschreiben die einzelnen Felder eines EAP Pakets näher.

A.1.1 Allgemeiner Paketaufbau

- **EAP-Code (1 Byte)**

Name	Wert	Beschreibung
REQUEST	1	Forderung nach Authentifizierungsdaten.
RESPONSE	2	Antwort mit Authentifizierungsdaten EAP-Identifier bildet Referenz zum REQUEST Paket.
SUCCESS	3	Authentifizierung erfolgreich.
FAILURE	4	Authentifizierung fehlgeschlagen.

- **EAP-Identifier (1 Byte)**
  - Eindeutiger Identifier. Dieser wird benötigt, um Wiederholungssendungen zu erkennen bzw. als eindeutiges Zuordnungsmerkmal für erhaltene RESPONSE Pakete.
- **EAP-Data Length (2 Byte)**
  - Beinhaltet die Länge des EAP-Paketes einschließlich der Headerdaten.
- **EAP-Data**
  - Nutzdaten des Paketes. Diese sollten nur bei EAP-REQUEST und EAP-RESPONSE vorhanden sein.

A.1.2 Authentifizierungsverfahren

Das Datenfeld während einer Authentifizierung besteht aus einem Typfeld und einem Datenfeld, welches abhängig von Authentifizierung eine unterschiedliche Struktur besitzt. Der Aufbau der Daten der Verfahren für die Authentifizierung kann in den Quellen nachgelesen werden.

- **Type (1 Byte)**

Name	Wert	Beschreibung
NAK	3	Antwort vom Supplicant wenn das Verfahren zur Authentifizierung nicht unterstützt wird.
MD5	4	MD5-Challenge
OTP	5	One-Time-Password
GTC	6	Generic-Token-Card
TLS	13	Transport Layer Security
TTLS	21	Tunneled Transport Layer Security
PEAP	25	Protected EAP

- **Daten**
  - Nutzdaten des Authentifozierungsverfahrens. Diese sind abhängig von dem gewählten Typ.

A.2 EAPOL

Die folgenden Abschnitte beschreiben die einzelnen Felder eines EAPOL Pakets näher.

A.2.1 Allgemeiner Paketaufbau

- **EAPOL-Version (1 Byte)**
  - Der Wert dieses Feldes ist momentan immer "1", da zur Zeit keine Folgeversion definiert ist.
- **EAPOL-Type (1 Byte)**

Name	Wert	Beschreibung
EAP	0	EAPOL Paket beinhaltet EAP Paket
START	1	Start einer EAPOL Verbindung
LOGOFF	2	Beendigung einer EAPOL Verbindung
KEY	3	EAPOL-Key Paket. Der Aufbau des Keypaketes wird im Abschnitt A.2.2 beschrieben.
ALERT	4	EAPOL Fehlermeldung

- **EAPOL-Data Length (2 Byte)**
  - Beinhaltet die Länge des EAPOL Paketes einschließlich der Headerdaten.
- **EAPOL-Data**
  - Nutzdaten des Paketes. Diese sollten nur bei EAPOL-EAP, EAPOL-ALERT und EAPOL-KEY vorhanden sein.

A.2.2 EAPOL-Key Paket

Nach erfolgter Authentifizierung erhält der Authenticator bei bestimmten Verfahren vom Authentication-Server Schlüsselinformationen. Diese werden im RADIUS Paket in der Form vom MPPE Attributen übermittelt. Nach erfolgreicher Entschlüsselung dieser Attribute kann der Session-Key und die Server-Key für die Erzeugung der EAPOL-Key Pakete genutzt werden. Informationen über die MPPE Attribute sind im Abschnitt A.3 enthalten. Die Schlüsselinformationen werden beim Software-Authenticator im DebugLevel "1" abhängig vom Supplicant ausgegeben. Die Nutzdaten des EAPOL-Key Paketes sind wie folgt aufgebaut.

- **Type (1 Byte)**
  - Momentan ist nur eine Art des Typs definiert. Der Schlüssel wird standardmäßig durch den RC4-Algorithmus gesichert, dass dieser nicht im Klartext übertragen wird.
  - Type = 1 (RC4)
- **Length (2 Byte)**
  - Beinhaltet die eigentliche Schlüssellänge in Byte, wobei der IV bei WEP vernachlässigt wird.
- **Replay Count (8 Byte)**
  - Beinhaltet einen NTP Zeitstempel um sicherzustellen, dass keine alten Schlüssel durch Angreifer erneut eingespielt werden.
- **Key Initialisation Vector (16 Byte)**
  - Beinhaltet 16 Byte Zufallswerte für den RC4-Schlüssel.
- **Key Index (1 Byte)**
  - Das Key Index Feld wird bitweise interpretiert.
  - Bit 0-6: Schlüsselnummer
  - Bit 7:
    - 0 : Multicast bzw. Group Key
    - 1 : Unicast Key
- **Key Signature (16 Byte)**
  - Signatur über das vollständige Paket, wobei die 16 Byte des Signaturfeldes auf null gesetzt werden.
  - Die Berechnung erfolgt über einen HMAC-MD5. Dafür wird der Server-Schlüssel, welcher bei der Authentifizierung generiert wurde, verwendet.
- **Key (Länge entspricht dem "Length-Feld")**
  - Falls ein vordefinierte Schlüssel übermittelt wird, befindet dieser sich durch den RC4 geschützt und der entsprechenden Längen an dieser Stelle des Pakets. Wird kein angegebener Schlüssel übermittelt, werden die ersten n Bytes des Session-Schlüssels verwendet.
  - Der Schlüssel, welcher für den RC4 genutzt wird, ist eine Kombination aus dem Key-IV dieses Pakets und dem Session-Schlüssel, welcher ebenso vom Authentication-Server übermittelt wird.

A.3 RADIUS

Die folgenden Abschnitte enthalten detaillierte Angaben über die beim 802.1x Standard benötigten Informationen zur Kommunikation mit dem RADIUS Server. Es werden Unterschiede zwischen gesendeten und Paketen, welche vom RADIUS Server empfangen werden, betrachtet.

A.3.1 Allgemeiner Aufbau

Die Kommunikation zwischen Authenticator und Authentication-Server erfolgt über UDP. Die Nutzdaten sind entsprechend dem RADIUS Protokoll wie folgt aufgebaut.

- **Code (1 Byte)**

Name	Wert	Beschreibung
ACCESS REQUEST	1	Anfragen für eine Authentifizierung.
ACCESS ACCEPT	2	Authentifizierung erfolgreich.
ACCESS REJECT	3	Authentifizierung fehlgeschlagen.
ACCESS CHALLENGE	11	Forderung nach weiteren Daten zur Authentifizierung.

- **Identifier (1 Byte)**
  - Entspricht bei ACCESS-REQUEST Paketen der ID des EAP-RESPONSE Paketes vom Supplicant. Bei allen anderen wird der empfangene Identifier inkrementiert.
- **Length (2 Byte)**
  - RADIUS Paket Länge einschließlich der Headerdaten.
- **Authenticator (16 Byte)**
  - Ein MD5 Hashwert von 16 Byte Zufallswerten.
- **Attribute**
  - Hier folgt eine beliebige Anzahl an RADIUS Attributen. Die für eine EAP Sitzung notwendigen werden in den folgenden Abschnitten vorgestellt.
  - Attributaufbau:
    - Attribut-Typ (1 Byte)
    - Attribut-Length (1 Byte)
      - Die Länge kann max. 255 Byte betragen. Ist ein Attribut länger, muss es in mehrere Teile zerlegt werden.
    - Attribut-Data
      - Die Datenlänge kann max. 253 Byte betragen, da 2 Byte für den Attributheader notwendig sind.

A.3.2 Attribute für EAP

Die eigentliche Informationen zur Authentifizierung werden in den Attributen übertragen. Die beim Software-Authenticator verwendeten und für 802.1x notwendigen, sollen folgend, abhängig ob diese gesendet oder empfangen werden, vorgestellt werden.

Vom RADIUS Server empfangen

STATE	24	Zustandsidikator, welcher bei Folgepaketen an den RADIUS Server zur Authentifizierung mit gesendet werden muss.
VENDOR	26	Beinhaltet die Schlüsselinformationen

Zum RADIUS Server gesendet

Name	Wert	Beschreibung
User-Name	1	Identität, welche vom Supplicant übertragen wurde.
NAS-IP-Address	4	IP-Adresse des Authenticators
NAS-Port	5	Physische Protnummer des zu authentifizierenden Supplicant. (Software-Authenticator standardmäßig "1")
Called-Station-Id	30	Identifier-String des Authenticator
Calling-Station-Id	31	Identifier-String des Supplicant
NAS-Port-Type	61	Entspricht der Art der Verbindung, welche zwischen Supplicant und Authenticator besteht. (Software-Authenticator standardmäßig "19", was einer 802.11 Verbindung entspricht, damit Daten zur Schlüsselgenerierung vom RADIUS Server geliefert werden).
EAP-Message	79	EAP Paket vom Supplicant gekapselt in evtl. mehrere Attribute.
Message-Authenticator	80	Ein Paket eindeutiger Hashwert. Wird abschließend durch eine Funktion (HMAC-MD5(Radius-Secret + Radius Paket);) erzeugt.

Vendor-Attribut

In diesem Attribut werden die Schlüssel übermittelt. Dieses Attribut kann von Herstellern explizit genutzt werden. Die Schlüssel werden als MPPE Schlüssel, welche durch das Radius-Secret gesichert sind übertragen. Die Erzeugung der Klartextschlüssel kann in [54] nachgelesen werden. Die Funktion *_8021x_radius_decryptkey* in der Datei **_8021x_radius.c** beinhaltet die Implementierung dieses Verfahrens. Das Vendor-Attribut für diese Information ist wie folgt aufgebaut.

- **Identifier (4 Byte)**
  - Entspricht dem Microsoft Identifier (Identifier = "311")
- **Type (1 Byte)**

Server-Key (Send-Key)	16	Server-Schlüssel zur Erzeugung des Signatureintrages beim EAPOL-Key Paket.
Session-Key (Receive-Key)	17	Session-Schlüssel zur Verschlüsselung eines vordefinierten Schlüssel bzw. der eigentliche Schlüssel. Diese wird entsprechend des Verfahrens (WEP/WAP/802.11i) interpretiert.

- **Length (1 Byte)**
  - Die Länge entspricht der Datenlänge + Längefeld + Typfeld.
- **Data**
  - Schlüsseldaten.

Anhang B

Software-Authenticator

B.1 Funktionsreferenz

Dies ist eine Übersicht über die Hauptfunktionen, Strukturen, ihrer Verwendung und der Lokalität ihrer Implementierung.

- **8021x_auth (MAIN)**
  - Konstanten- & Variablendeklaration
    - struct _8021x_auth_authenticator
      - Speichert während der Ausführung globale Parameter des Authenticator.
- **8021x_config**
  - Funktionsdeklaration
    - int _8021x_config_readConfigFile
      (const char *filename, struct _8021x_auth_authenticator *theauth);
      - Liest eine Konfigurationsdatei ein und speichert die Werte in der globalen Struktur des Authenticators.
- **8021x_debug**
  - Konstanten- & Variablendeklaration
    - enum DEBUGSTATE
      {DEBUG=0,ERROR=1,WARNING=2,LOG=3,PACKET=4};
      - Liste der verschiedenen Augabevarianten
    - struct DEBUGSTRUCT
      - Speichert Information über die einzelnen Ausgabemöglichkeiten.

  - Funktionsdeklaration
    - void _8021x_debug(enum DEBUGSTATE output, const char *format, ...);
      - Gibt eine Debugmeldung eines bestimmten Typs aus.

- **8021x_eap**
  - Konstanten- & Variablendeklaration
    - struct _8021x_eap_header
      - Headerdaten eines EAP-Paketes
    - struct _8021x_eap_pkt
      - EAP Paket
  - Funktionsdeklaration
    - int _8021x_eap_identity(unsigned short int id, unsigned char **packet);
      - Erzeugt ein EAP-Request Identity Paket.
    - int _8021x_eap_success(unsigned short int id, unsigned char **packet);
      - Erzeugt ein EAP-Success Paket (erfolgreiche Authentifizierung).
    - int _8021x_eap_failure(unsigned short int id, unsigned char **packet);
      - Erzeugt ein EAP-Failure Paket (Authentifizierung fehlgeschlagen).

- **8021x_eapol**
  - Konstanten- & Variablendeklaration
    - struct _8021x_eapol_header
      - Headerdaten eines EAPOL Pakets
    - struct _8021x_eapol_pkt
      - EAPOL-Paket
    - struct _8021x_eapol_key_descriptor
      - EAPOL-Key Descriptor Paket Daten
  - Funktionsdeklaration
    - int _8021x_eapol_pktbuild(unsigned int pkttype, unsigned char *data, unsigned int datalen, unsigned char **packet);
      - Erzeugt EAPOL Pakete.

- **8021x_ethernet**
  - Konstanten- & Variablendeklaration
    - struct _8021x_ethernet_header
      - Headerdaten eines Ethernet-Pakets

    - struct _8021x_ethernet_hpkt
        - Ethernet-Pakets
  - Funktionsdeklaration
    - int _8021x_ethernet_pktbuild
      (unsigned char *dstaddr, unsigned char *srcaddr, unsigned short int protocol, unsigned char *data, unsigned int datalen, unsigned char **packet);
        - Erzeugt Ethernet-Pakete mit einer übergebenen Nachricht.

- **8021x_filter**
  - Funktionsdeklaration
    - int _8021x_filter_init(unsigned char *filterprog);
        - Initialisiert durch Aufruf eines externen Programms die Filterfunktionen.
    - int _8021x_filter_deinit(unsigned char *filterprog);
        - Beendet die Funktion des Filters.
    - int _8021x_filter_addRule
      (struct _8021x_supp_supplicant *supp, unsigned char *filterprog);
        - Fügt einen Supplicant in die Filterliste als authentifiziert ein.
    - int _8021x_filter_delRule
      (struct _8021x_supp_supplicant *supp, unsigned char *filterprog);
        - Löscht einen Supplicanten aus der Filterliste

- **8021x_global**
  - Konstanten- & Variablendeklaration
    - Hier werden globale Konstanten für alle Programmteile definiert. Diese Konstanten sind an der Verwendung von ausschließlich Großbuchstaben zu erkennen.
  - Funktionsdeklaration
    - int _8021x_global_ntp_timestamp(int *ntp_hi, int *ntp_lo);
        - Erzeugt einen dem NTP Format entsprechenden Zeitstempel.
    - int _8021x_global_getRandoms(unsigned char *buffer, unsigned int size);
        - Füllt einen Buffer mit Zufallszahlen.
    - int _8021x_global_ConvertKey(unsigned char *key, unsigned short int keylen);
        - Convertiert das Master-Secret, welches vom RADIUS Server übermittelt wurde.
    - unsigned char _8021x_global_mkId(unsigned char *id);
        - Erzeugt eindeutige ID's für die EAP Pakete, um Konflikte zwischen den Supplicanten zu vermeiden.

- **8021x_nal**
  - Konstanten- & Variablendeklaration
    - struct _8021x_nal_descriptor
      - Hält Daten über das Interface, mit welchem sich die Supplicanten verbinden.
  - Funktionsdeklaration
    - int _8021x_nal_initialize(struct _8021x_nal_descriptor *naldesc);
      - Initialisiert das Interface.
    - int _8021x_nal_close (struct _8021x_nal_descriptor *desc);
      - Beendet die Nutzung des Interfaces und gibt die Ressourcen frei.
    - int _8021x_nal_send
      (struct _8021x_nal_descriptor *desc, unsigned char *packet, unsigned int length);
      - Sendet ein Paket über das Interface.
    - int _8021x_nal_capture_loop
      (struct _8021x_nal_descriptor *naldesc, unsigned char *buffer);
      - Empfängt Pakete über das Interface.
- **8021x_radius**
  - Konstanten- & Variablendeklaration
    - struct _8021x_radius_descriptor
      - Beinhaltet Informationen über die Verbindung zum RADIUS Server
    - struct _8021x_radius_header
      - RADIUS Paketheader
    - struct _8021x_radius_attr
      - RADIUS Attributstruktur
    - struct _8021x_radius_pkt
      - RADIUS Paket
    - struct _8021x_radius_build
      - Enthält Werte zum Erzeugen von RADIUS Paketen
  - Funktionsdeklaration
    - int _8021x_radius_init(struct _8021x_radius_descriptor *radiusdesc);
      - Öffnet die Verbindung zum RADIUS Server.
    - int _8021x_radius_close (struct _8021x_radius_descriptor *radiusdesc);
      - Beendet die Verbindung zum RADIUS Server.

* int _8021x_radius_send (struct _8021x_radius_descriptor *radiusdesc, unsigned char *packet, unsigned short int pktlen);
  - Sendet RADIUS Pakete.
* int _8021x_radius_recv (struct _8021x_radius_descriptor *radiusdesc, unsigned char *buffer, unsigned int buflen);
  - Empfängt RADIUS Pakete.
* int _8021x_radius_pktbuild (struct _8021x_radius_descriptor *raddesc, struct _8021x_radius_build *build, unsigned char code, unsigned char **packet);
  - Erzeugt RADIUS Pakete.
* int _8021x_radius_msgauthenticator (unsigned char *secret, unsigned short int seclen, unsigned char *radpkt, unsigned short int pktlen, unsigned char *msgauthenticator);
  - Erzeugt einen Messageauthenticator (HMAC).
* int _8021x_radius_keyinformation (unsigned char *radvalue, struct _8021x_radius_descriptor *raddesc, struct _8021x_supp_supplicant *supp);
  - Ermittelt das Master-Secret aus dem übermittelten MPPE Schlüssel und trägt dieses in die Supplicantenstruktur ein.

- **8021x_rx**
  - Funktionsdeklaration
    * int _8021x_rx_ethernet_pktparse(unsigned char *pktbuffer, unsigned char auth_ethernet_addr[_8021X_ETHADDR_LEN_], struct _8021x_supp_supplicant **supp);
      - Analysiert ein erhaltenes Ethernet-Paket und setzt entsprechende Werte in der Supplicantenstruktur.
    * int _8021x_rx_eapol_pktparse (unsigned char *packet,struct _8021x_auth_authenticator *theauth, struct _8021x_supp_supplicant **supp);
      - Analysiert ein erhaltenes EAPOL Paket und setzt entsprechende Werte in der Supplicantenstruktur.
    * int _8021x_rx_eap_pktparse (unsigned char *packet, struct _8021x_supp_supplicant **supplicants);
      - Analysiert ein erhaltenes EAP Paket und setzt entsprechende Werte in der Supplicantenstruktur.

        * int _8021x_rx_radius_pktparse(struct _8021x_radius_descriptor *raddesc, struct _8021x_supp_supplicant *supplicants, unsigned char *packet,unsigned int pktlen);
            · Analysiert ein erhaltenes RADIUS Paket und setzt entsprechende Werte in der Supplicantenstruktur.

- **8021x_sm**
    - Konstanten- & Variablendeklaration
        * enum _8021X_STATE_
            · Definiert Aliasnamen für die einzelnen Zustandsnummern
    - Funktionsdeklaration
        * int _8021x_sm_auth(struct _8021x_auth_authenticator *theauth, struct _8021x_supp_supplicant *supplicants, struct _8021x_radius_descriptor *raddesc, struct _8021x_nal_descriptor *naldesc);
            · Authenticator State Machine
        * int _8021x_sm_bauth(struct _8021x_auth_authenticator *theauth, struct _8021x_supp_supplicant *supplicants, struct _8021x_radius_descriptor *raddesc, struct _8021x_nal_descriptor *naldesc);
            · Backend-Authentication State Machine
        * int _8021x_sm_rauth(struct _8021x_auth_authenticator *theauth, struct _8021x_supp_supplicant *supplicants, struct _8021x_radius_descriptor *raddesc, struct _8021x_nal_descriptor *naldesc);
            · Reauthentication State Machine
        * int _8021x_sm_key(struct _8021x_auth_authenticator *theauth, struct _8021x_supp_supplicant *supplicants, struct _8021x_radius_descriptor *raddesc, struct _8021x_nal_descriptor *naldesc);
            · Keytransmit State Machine

- **8021x_supp**
    - Konstanten- & Variablendeklaration
        * struct _8021x_supp_supplicant
            · Teil der dynamischen Datenstruktur. Hält Informationen über die einzelnen Supplicanten.
    - Funktionsdeklaration
        * void _8021x_supp_init(struct _8021x_supp_supplicant **supp);
            · Initialisiert die Datenstruktur.

    - int _8021x_supp_isempty(struct _8021x_supp_supplicant **supp);
        - Prüft ob sich Supplicanten in der Liste befinden.
    - int _8021x_supp_empty(struct _8021x_supp_supplicant **supp);
        - Löscht alle Supplicanten.
    - int _8021x_supp_add
      (struct _8021x_supp_supplicant **supp, unsigned char *ethaddr);
        - Fügt einen Supplicanten der List hinzu.
    - int _8021x_supp_del(struct _8021x_supp_supplicant **supp, unsigned char *ethaddr);
        - Löscht einen Supplicanten aus der Liste.
    - struct _8021x_supp_supplicant *_8021x_supp_find
      (struct _8021x_supp_supplicant **supp, unsigned char *ethaddr);
        - Sucht einen Supplicanten anhand der MAC-Adresse.
    - struct _8021x_supp_supplicant *_8021x_supp_findid
      (struct _8021x_supp_supplicant *supp, unsigned char id);
        - Sucht einen Supplicanten anhand der EAP Paket ID.
    - struct _8021x_supp_supplicant *_8021x_supp_next
      (struct _8021x_supp_supplicant *supp);
        - Liefert den nächsten Supplicanten der Liste.
    - int _8021x_supp_initialValues
      (struct _8021x_supp_supplicant *supp, unsigned char portControl,
      unsigned char portEnabled, unsigned short int currentId);
        - Setzt Initialisierungswerte für einen Supplicanten.
    - int _8021x_supp_gCollection
      (struct _8021x_supp_supplicant **supplicants, unsigned char *filterprogram);
        - Sucht nach nicht mehr benötigten Supplicanten, löscht diese und entfernt alle Filterregeln für diesen.

- **8021x_timer**
    - Wurde als Sekundentimer genutzt. Dient jetzt der Signalbehandlung.
    - Funktionsdeklaration
        - int _8021x_timer_init();
            - Trägt einen Funktion zur Behandlung verschiedener Signale ein.

- **8021x_tx**
  - Funktionsdeklaration
    - int _8021x_tx_CannedFail
      (struct _8021x_nal_descriptor *naldesc, struct _8021x_supp_supplicant *supp);
      - Sendet den Fehlschlag der Authentifizierung.
    - int _8021x_tx_CannedSuccess
      (struct _8021x_nal_descriptor *naldesc, struct _8021x_supp_supplicant *supp);
      - Sendet den Erfolg der Authentifizierung.
    - int _8021x_tx_RadResponse
      (struct _8021x_nal_descriptor *naldesc, struct _8021x_supp_supplicant *supp);
      - Sendet ein gekapseltes EAP Paket vom RADIUS Server an den Supplicant.
    - int _8021x_tx_Identity
      (struct _8021x_nal_descriptor *naldesc, struct _8021x_supp_supplicant *supp);
      - Sendet ein EAP-Request Identity Paket an den Supplicant.
    - int _8021x_tx_sendRespToServer
      (struct _8021x_radius_descriptor *raddesc, struct _8021x_supp_supplicant *supp);
      - Sendet EAP Paket vom Supplicant an den RADIUS Server.
    - int _8021x_tx_key(struct _8021x_auth_authenticator *theauth,
      struct _8021x_supp_supplicant *supp, struct _8021x_nal_descriptor *naldesc,
      unsigned int index, unsigned int broadcast);
      - Sendet EAPOL-Key Pakete an den Supplicant.

B.2 Konfigurationsbeispiel

Dieses Listing stellt einen Bespielkonfiguration für den Software-Authenticator dar. Die Bedeutung der einzelnen Elemente ist der Tabelle 6.3, auf Seite 59 zu entnehmen.

"8021x_auth.conf"

```
# Beispielkonfiguration des Software-Authenticator

# Debug Level
DebugLevel="1"

# Logdatei
LOG="/tmp/8021x.log"

# Debug/Error/Warning Datei
DEBUG=""
ERROR=""
WARNING=""

# Packet dump Datei
PACKET=""

# Interface
Device="eth1"
PromiscousMode="1"
CaptureFilter="ether proto 0x888e"

# RADIUS
RadiusIP="127.0.0.1"
RadiusPort="1812"
RadiusSecret="testing123"

CalledStationId="8021x Software Authenticator"
CallingStationId="8021x Supplicant"

# Statemachine
ReAuthenticate="1"
PortControl="AUTO"

SupplicantRequestTime="10"
RadiusRequestTime="10"
ReAuthenticateTime="120"

# Filter
FilterProgram="./8021x_wlan.pl"

# Wireless Key Information (WEP)
KeyTxEnabled="0"
Key="1122334455"
KeyLength="5"
```

Anhang C

Supplicanten Konfiguration

C.1 Open1x

Der *xsupplicant* von Open1x wurde für die Nutzung von 802.1x unter Linux entwickelt. Für die Nutzung von Verfahren, welche eine TLS-Verbindung herstellen, ist es wichtig, die nötigen Zertifikate und Informationen zu besitzen. Dazu gehören die RootCA und zusätzlich für TLS das Zertifikat für den Supplicant. Eine mögliche Konfiguration für TLS und TTLS mit PAP, zeigt folgendes Listing.

"xsupplicant.conf"

```
# xsupplicant config

default_netname = default
network_list = all

# Dieses Kommando wird nach der Authentifizierung ausgefuehrt.
first_auth_command = <BEGIN_COMMAND>/sbin/dhclient %i<END_COMMAND>

# Wird nach einer Reauthentifizierung ausgefuehrt.
#reauth_command = <BEGIN_COMMAND>...<END_COMMAND>

# Wird beim Start ausgefuehrt.
#startup_command = <BEGIN_COMMAND>...<END_COMMAND>

# logging
logfile = /var/log/xsupplicant.log

default {
    # authentication (ALL, EAP-MD5, EAP-TLS, EAP-TTLS, ...)
    allow_types = all

    # Destination MAC-Address (standardmaessig: 01:80:C2:00:00:03)
    #dest_mac = 00:00:CB:53:22:A5

    # aeussere Identitaet bei TTLS und PEAP, ansonsten Hauptidentitaet
    identity = <BEGIN_ID>ril<END_ID>
```

```
    # Wichtig fuer das Verwendung von dynamischen Schluesseln.
    # Werte: "wired" und "wireless"
    type = wireless

    # Sollen die Schlussel gesetzt werden? (yes/no)
    wireless_control = yes

    eap-tls {

        # Supplicant Zertifikat
        user_cert = supp.pem
        # Supplicant Schluessel
        user_key = supp.pem
        # Zertifikat Passwort
        user_key_pass = <BEGIN_PASS>whatever<END_PASS>
        # RootCA (Muss das Authentication-Server Zertifikat ebenso
        # zertifiziert haben.
        root_cert = root.pem

        # EAP Zusatzoptionen
        chunk_size = 1398
        crl_dir = revoked
        random_file = /dev/urandom
        session_resume = no

        # Verifizierung des CN-Parameters im Authentication-Server
        # Zertifikat. Ist nicht unbedingt notwendig, erhoeht jedoch
        # die Sicherheit vor Man-in-the-Middel Angriffen.
        cncheck = stargate.home.de
        # yes check complete, no check domain
        cnexact = yes

    }

    eap-ttls {
        # RootCA - Zur Verifizierung des Authentication-Server Zertifikates
        root_cert = root.pem

        # Inneres Protokoll und Authentifizierungsparameter
        phase2_type = pap
        pap {
            username = <BEGIN_UNAME>ril<END_UNAME>
            password = <BEGIN_PASS>rilril<END_PASS>
        }

        # EAP Zusatzoptionen
        chunk_size = 1398
        random_file = /dev/urandom
    }
}
```

C.2 Wire1x

Der Supplicant *Wire1x* von Wireless Internet Research & Engineering (WIRE) Lab. stellt die Funktionalität für alle Windowssysteme zur Verfügung. Wichtig bei der Nutzung dieser Software ist, dass die neusten Versionen der "LIBNET", "LIBPCAP" und die Bibliotheken von "OpenSSL" in Programmverzeichnis existieren, oder global installiert sind. Zusätzlich muss für die Verwendung von TLS, PEAP und TTLS die für die Verifizierung des Zertifikates des Authentication-Server nötige RootCA als **PEM** [55] Datei existieren. Bei der Verwendung von TLS ist es zusätzlich notwendig das Zertifikat und den Schlüssel des Supplicanten zu importieren. Dazu wird das Zertifikat in einer Datei im **PEM** Format in das Programmverzeichnis kopiert und der nötige Schlüssel mit Hilfe einer **PCSK#12** [55] Datei unter Windows importiert. Für die Nutzung unter Windows muss den Zertifikaten ein besonderer Parameter beigefügt sein, welcher für die Verwendung von 802.1x notwendig ist. Der Parameter *Server/Client Authentication Enhanced Key Usage (EKU)* besitzt einen entsprechenden Identifier, welcher der folgenden Tabelle entnommen werden kann.

Server Authentication EKU	1.3.6.1.5.5.7.3.1
Client Authentication EKU	1.3.6.1.5.5.7.3.2

Die folgende Abbildung C.1 zeigt das Programmfenster für das TTLS-Verfahren. Die **Unprotected ID** entspricht der äußeren Identität, welche zum Aufbau der TLS-Verbindung genutzt wird. Die Felder **Username** und **Password** sind selbst erklärend. Anschließend muss die Auswahl des inneren Protokolls erfolgen. Beim der Verwendung dieser Software an der TU Chemnitz wird die Einstellung PAP gewählt. Als letztes erfolgt die Auswahl des Interfaces über welches die Authentifizierung durchgeführt werden soll.

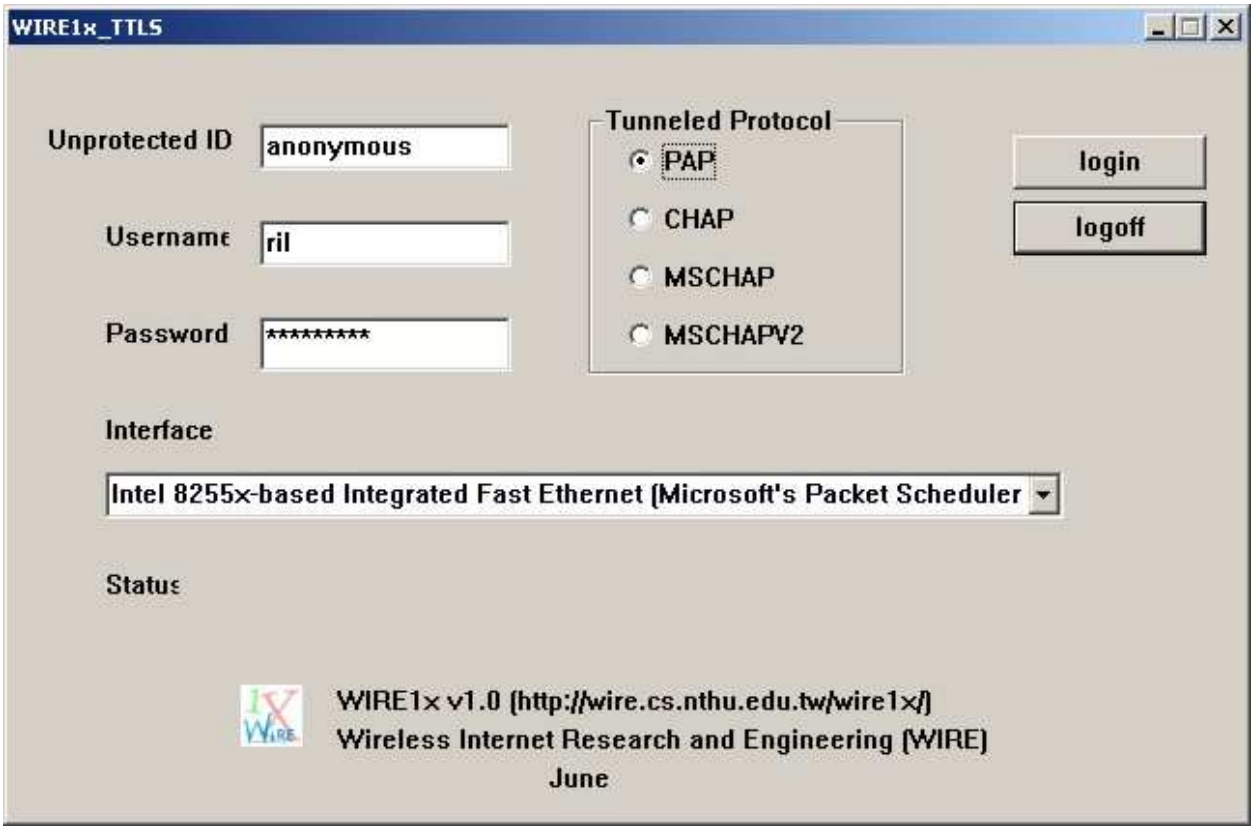

Abbildung C.1: Wire1x-TTLS Supplicant (ohne Patch)

Um den Einsatz in einem durch Switchtechnologien realisierten Netzwerk zu gewährleisten, wurde im Rahmen dieser Arbeit ein Patch für die TTLS-Variante von *Wire1x* implementiert. Die Abbildung C.2 zeigt den neuen Aufbau des Programmfensters. Der einzige Unterschied ist das Eingabefeld für eine MAC-Adresse. Diese sollte die MAC-Adresse des Authentication-Servers sein. Die geänderten Dateien für diesen Patch, befinden sich auf der beigefügten CD. Das Statusfeld zeigt während der Authentifizierung den aktuellen Zustand der Statemachine des Supplicanten an.

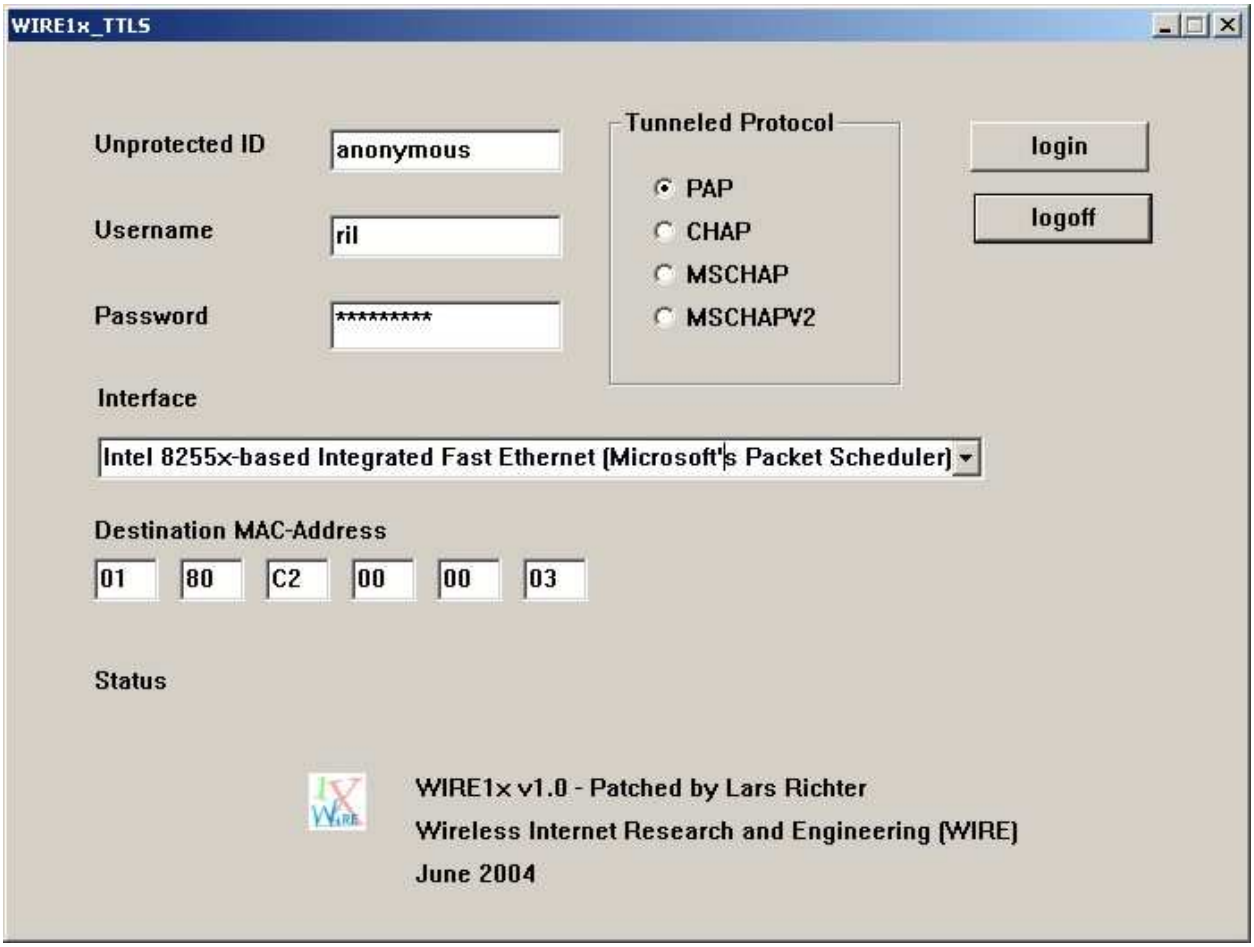

Abbildung C.2: Wire1x-TTLS Supplicant (mit Patch)

Anhang D

RADIUS Konfiguration

Das folgende Listing zeigt einen Auszug aus der Konfiguration des FreeRADIUS. Die dargestellten Eigenschaften bewirken eine Authentifizierung mittels 802.1x unter Verwendung von TTLS. Im TLS gesicherten Kanal wird anschließend das Verfahren PAP genutzt. Die Prüfung der Authentifizierung erfolgt in diesem Beispiel anhand der Passworteinträge in der Datei "*/etc/passwd*". Der RADIUS Server bietet jedoch weitaus mehr Möglichkeiten an. So sind ebenfalls Module für die Verwendung von LDAP, SAMBA, Kerberos, SQL-Datenbanken und PAM vorhanden. Die Nutzung dieser ist ebenfalls mit EAP-TTLS und PAP möglich. Durch die Verwendung von Klartextpasswörtern bei PAP kann eine Prüfung gegen alle Verfahren vorgenommen werden.

"radius.conf"

```
# Module Configuration Section
modules {

    # PAP module to authenticate users based on their stored password
    #
    #  Supports multiple encryption schemes
    #  clear: Clear text
    #  crypt: Unix crypt
    #    md5: MD5 ecnryption
    #   sha1: SHA1 encryption.
    #  DEFAULT: crypt
    pap {
        encryption_scheme = clear
    }

    # EAP Module for 802.1x Authentification
    eap {

        ## Standard EAP Authentifizierungsverfahren
        default_eap_type = ttls

        ## EAP-TLS - Diese Einstellungen werden ebenso üfr TTLS öbentigt.
        tls {
```

```
            # Server-üSchlssel mit Passwort
            private_key_password = whatever
            private_key_file = ${raddbdir}/certs/server.pem

            # Serverzertifikat
            certificate_file = ${raddbdir}/certs/server.pem

            #  Trusted Root CA list
            CA_file = ${raddbdir}/certs/root.pem

            dh_file = ${raddbdir}/certs/dh
            random_file = ${raddbdir}/certs/random

            fragment_size = 1024

            include_length = yes
        }

        # EAP-TTLS
        ttls {
            # Inneres Authentifizierungsverfahren
            # Zur Zeit ist von EAP nur EAP-MD5 als solches verwendbar.
            # Jedoch koennen auch EAP fremde Verfahren wie PAP verwendet werden.
            default_eap_type = md5

            copy_request_to_tunnel = no

            use_tunneled_reply = no
        }
    }

    unix {
        cache = yes

        cache_reload = 600

        passwd = /etc/passwd
        shadow = /etc/shadow
        group = /etc/group

        radwtmp = ${logdir}/radwtmp
    }

}

#  Authorization - Section
authorize {

    #  This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
    eap

}

#  Authentication Section
authenticate {
```

```
    # PAP authentication, when a back-end database listed
    # in the 'authorize' section supplies a password. The
    # password can be clear-text, or encrypted.
    Auth-Type PAP {
        pap
    }

    # Authentifizierung anhand von /etc/passwd
    unix

    # Allow EAP authentication.
    eap

}
```

Anhang E

CD Inhalt

Verzeichnisstruktur

- "8021x_auth"
  - "conf"
    - Konfigurationsdatei für den Software-Authenticator
  - "filter"
    - Filterprogramme für den Einsatz an der TU Chemnitz sowie ein Beispielskript
  - "include"
    - Includedateien des Software-Authenticators
  - "src"
    - Sourcecodedateien des Software-Authenticators
- "doc"
  - Diese Dokumentation!
- "mib"
  - MIB für die Nutzung eines SNMP Interfaces eines Hardwareauthenticators.
- "supplicant"
  - "wire1x"
    - Geänderte Dateien für die Nutzung einer Unicast-Adresse bei TTLS.
  - "xsupplicant"
    - Beispielkonfiguration für die Nutzung des *xsupplicant* von Open1x

Literaturverzeichnis

[1] J. Postel, J. Reynolds
RFC 959 - File Transfer Protocol, Oktober 1985

[2] T. Dierks, C. Allen
RFC 2246 - The TLS Protocol Version 1.0, Januar 1999

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee
RFC 2616 - Hypertext Transfer Protocol, Juni 1999

[4] M. Crispin
RFC 3501 - Internet Message Access Protocol, März 2003

[5] ANSI/IEEE Std 802.11
Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements –
Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Stand 1999

[6] Matthew S. Gast
802.11 Wireless Networks - The Definitive Guide
O'Reilly, 1. Auflage, April 2002

[7] Prof. Dr.-Ing. habil. Uwe Hübner
Wireless Local Area Networks, April 2003
http://rnvs.informatik.tu-chemnitz.de/wlan/index.htm

[8] Bruce Potter & Bob Fleck
802.11 Security
O'Reilly, 1. Auflage, Dezember 2002

[9] Joseph Davies
Deploying Secure 802.11 Wireless Networks with Microsoft Windows
Microsoft Press, 1. Auflage, 2003

[10] Alex Hagedorn
IEEE 802.11i Sicheheit in drahtlosen lokalen Netzen
Diplomarbeit, Fachgebiet Kryptographie & Computeralgebra, TU Darmstadt , Novenber 2003

[11] Adam Subblefield, John Ioannidis, Aviel D. Rubin
Using the Fluhrer, Mantin and Shamir Attack to Break WEP
http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/stubbl.pdf

[12] Markus Nispel
Schluss mit löchrig - Sichere Wireless LANs mit IEEE 802.11i, 2004
http://www.kes.info/archiv/online/04-5-036.htm

[13] Wikipedia
OSI-Modell
http://de.wikipedia.org/wiki/OSI-Modell

[14] S. Kent, R. Atkinson
RFC 2401 - Security Architecture for the Internet Protocol, November 1998

[15] Mirko Parthey
IP Security für Linux, Studienarbeit, Fakultät für Informatik, TU Chemnitz, September 2000
http://archiv.tu-chemnitz.de/pub/2001/0008/

[16] IEEE Std 802.1X
IEEE Standard for Local and metropolitan area networks –
Port-Based Network Access Control, Stand 2001

[17] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn
RFC 2637 - Point-to-Point Tunneling Protocol (PPTP), Juli 1999

[18] Johannes Helmig
Virtual Private Networks (VPN / PPTP), Juli 1998
http://www.wown.com/j_helmig/vpn.htm

[19] W. Simpson
RFC 1661 - The Point-to-Point Protocol (PPP), Juli 1994

[20] IEEE Std 802.1D
IEEE Standard for Local and metropolitan area networks
Media Access Control (MAC) Bridges, Stand 2004

[21] L. Blunk & J. Vollbrecht
RFC 2284 - PPP Extensible Authentication Protocol (EAP), März 1998

[22] R. Rivest
RFC 1321 - The MD5 Message-Digest Algorithm, April 1992

[23] B. Aboba, D Simon
RFC 2716 - PPP EAP TLS Authentication Protocol, Oktober 1999

[24] B. Lloyd, W. Simpson
RFC 1334 - PPP Authentication Protocols, Oktober 1992

[25] W. Simpson
PPP Challenge Handshake Authentication Protocol (CHAP), August 1996

[26] Ashwin Palekar, Dan Simon, Joe Salowey, Hao Zhou, Glen Zorn, S. Josefsson
Protected EAP Protocol (PEAP) Version 2, Oktober 2004
http://www.ietf.org/internet-drafts/draft-josefsson-pppext-eap-tls-eap-10.txt

[27] Symbol Technologies, Inc.
Protected Extensible Authentication Protocol (PEAP), Oktober 2003
http://www.symbol.com/products/wireless/peap.html

[28] Cameron Macnally
Cisco LEAP protocol description, September 2001
http://lists.cistron.nl/pipermail/cistron-radius/2001-September/002042.html

[29] Cisco Systems, Inc.
Dictionary Attack on Cisco LEAP Vulnerability, Stand Juli 2004
http://www.cisco.com/en/US/tech/tk722/tk809/technologies_security_notice09186a00801aa80f.html

[30] J. Arkko, H. Haverinen
Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA), April 2004
http://www.ietf.org/internet-drafts/draft-arkko-pppext-eap-aka-12.txt

[31] H. Haverinen, J. Salowey
Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM), April 2004
http://www.ietf.org/internet-drafts/draft-haverinen-pppext-eap-sim-13.txt

[32] Jouni Malinen
Host AP driver for Intersil Prism2/2.5/3, Oktober 2004
http://hostap.epitest.fi

[33] Greg Chesson, Henry Qian, Kevin Yu, Mathieu Lacage, Michael Renzmann, Sam Leffler, Stephane Laroche, William S. Kish

Multiband Atheros Driver for WiFi (MADWIFI), September 2004
http://sourceforge.net/projects/madwifi/

[34] Alfa & Ariss
SecureW2, Version 2.2
http://www.securew2.com/uk/

[35] Wireless Internet Research & Engineering (WIRE) Lab.
Wire1x, Juni 2003
http://wire.cs.nthu.edu.tw/wire1x/

[36] Arunesh Mishra, Nick L. Petroni, Jr., Bryan D. Payne, Chris Hessing, Terry Simons
Open1x, Version 1.0, Juni 2004
http://www.open1x.org

[37] Jean Tourrilhes
Wireless Tools for Linux, Mai 2004,
http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

[38] Jouni Malinen
Linux WPA/WPA2/IEEE 802.1X Supplicant, November 2004
http://hostap.epitest.fi/wpa_supplicant/

[39] Academic Computing and Communications Center
MacOS X Panther 802.1x Supplicant
http://www.uic.edu/depts/accc/network/wireless/macx.html

[40] Funk Software Inc.
Odyssey Client
http://www.funk.com/radius/wlan/wlan_c_radius.asp

[41] Meetinghouse Data Communications
AEGIS Client,
http://www.mtghouse.com/products/aegisclient/index.shtml

[42] C. Rigney, A. Rubens, W. Simpson, S. Willens
RFC 2138 - Remote Authentication Dial In User Service (RADIUS), April 1997

[43] The FreeRADIUS Project
FreeRADIUS Server, 2004
http://www.freeradius.org/

[44] Jonathan Hassel
RADIUS, O'Reilly, 1. Auflage, Oktober 2002

[45] IEEE Std 802.1Q
IEEE Standards for Local and metropolitan area networks
Virtual Bridged Local Area Networks, Stand 2003

[46] André Breiler
Differenzierte Bereitstellung von Internetdiensten in öffentlichen Bereichen der Universität
Diplomarbeit, Lehrstuhl für Rechnernetze und verteilte Systeme, Technische Universität Chemnitz, September 2000
http://archiv.tu-chemnitz.de/pub/2001/0009

[47] Technische Universität Chemnitz
Wlan-Funknetz MoCa - Mobiler Campus
http://www.tu-chemnitz.de/urz/netz/wlan/index.html

[48] Mirko Parthey
Zugangsmanagement für Wireless LAN
Diplomarbeit, Professur Rechnernetze und verteilte Systeme, Technische Universität Chemnitz, Dezember 2001
http://archiv.tu-chemnitz.de/pub/2002/0106

[49] Universitätsrechenzentrum - Techinsche Universität Chemnitz
Zertifikat der Internetnutzung (ZIN),
http://www.tu-chemnitz.de/urz/ZIN

[50] Rusty Russell
Linux 2.4 Packet Filtering HOWTO, 2002
http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html

[51] Mike Schiffman
The Evolution of Libnet, Februar 2004
http://www.packetfactory.net/Projects/Libnet/2004_RSA/eol-1.0.pdf

[52] Martin Casado
Packet Capture With libpcap and other Low Level Network Tricks
http://www.cet.nau.edu/ mc8/Socket/Tutorials/section1.html

[53] B. Aboba, P. Calhoun
RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP), September 2003

[54] G. Zorn
Microsoft Vendor-specific RADIUS Attributes, März 1999

[55] Nick Burch
Certificate Management with OpenSSL - General Stuff, November 2004
http://tirian.magd.ox.ac.uk/ nick/openssl-certs/general.shtml